

POLITECNICO DI MILANO
Corso di Laurea Magistrale in Automation and Control Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria



MPC CONTROL OF AN AUTONOMOUS WHEELCHAIR CONSIDERING RIDE COMFORT

Supervisor: Prof. Luca Bascetta

MSc Thesis of:
Adrià Sánchez, identification number 904320

Academic year 2017-2018

Abstract

This thesis work concerns vehicle robotics, the main objective being the development of an algorithm for trajectory planning for autonomous vehicles. To fulfill these requirements, the technique utilized is the so-called *Model Predictive Control*, which allows to perform an optimal trajectory generation while guaranteeing other control requisites such as obstacle avoidance, comfort or safety.

The focus is to assure ride comfort for the passenger in an scenario of obstacle avoidance, based on previous work on the system.

In this thesis, the vehicle is an electric wheelchair that receives reference values of velocity from a controller, which applying the MPC technique minimizes a suitable cost function based on the system and the geometry of the environment for a finite prediction horizon.

The requirements of ride comfort while fulfilling obstacle avoidance are based on previous study of human body vibrations in the literature, and their effect to the comfort sensation of the passenger, emphasizing in determining this sensation through an objective method.

The performance of the resulting control algorithm is evaluated by means of simulations to prove the effectiveness of this approach to the problem.

Contents

Abstract	1
1 Introduction	5
1.1 Motivation	5
1.2 Structure of the thesis	6
2 State of art	7
2.1 Experimental devices	7
2.2 Modelling of a comfort map based on subjective experiments	8
2.3 Suppression of discomfort vibrations	10
3 Model predictive control	15
3.1 System model	16
3.2 MPC basic formulation	17
3.3 Stability	18
3.4 Recursive feasibility	18
3.5 Definition of the optimization problem	19
4 System description	21
4.1 Kinematic model	21
4.2 Feedback linearization	22
4.3 Discrete-time model	24
5 Obstacle avoidance	25
5.1 Obstacle detection and segmentation	25
5.2 Constraints on state variables	27
5.3 Convexification of the admissible region	27
5.4 Definition of the position constraints	29
6 Ride comfort	33
6.1 Measurement and evaluation of vibration	33
6.2 Frequency weighting of the acceleration	35
6.3 Discretization and implementation of the weighting filter	38
7 MPC application	41
7.1 Cost function	42
7.2 Calibration of the controller	44
7.3 Constraints	45

7.3.1	Velocity constraints	45
7.3.2	Position constraints	46
7.3.3	Ride comfort constraints	48
7.4	Vortex Field	54
7.4.1	Application of the vortex field in the MPC problem	54
7.5	Activation and calculation of the vortex field vector	56
8	Simulations	61
8.1	Common parameters in the simulations	62
8.2	Convergence towards a reference	62
8.3	Verification of the position constraints	64
8.4	Simple obstacle avoidance	66
8.5	Multiple object avoidance	69
9	Conclusions and future developments	71

Chapter 1

Introduction

The work carried out in this thesis is part of the field of mobile robotics, a branch of robotics that deals with the design of robots that are able to move in the environment around them. This field also studies the design of autonomous vehicles, also known as *AGV* (*Automated Guided Vehicle*), which is the scope of this thesis. An autonomous vehicle is capable of navigating an uncontrolled environment with little to no need for physical guiding devices.

The components of a mobile robot are :

- **Controller:** generally a microprocessor, embedded microcontroller or a personal computer.
- **Control software:** it ranges in a wide variety of languages, from low-level to high-level programming.
- **Sensors:** used to conduct measures of the environment, thus the type of sensor used depends on the requirements of the robot.

The focus of this work is the control software, specifically the trajectory planning, i.e. the generation of the path the robot has to follow considering the obstacles detected in the vicinity. To achieve this, a technique called *Model Predictive Control* is utilized, which allows to find an optimal trajectory between the position of the robot and the reference while being able to easily define constraints on it.

1.1 Motivation

Parting from the work developed in [1], the main goal of this thesis is to improve the comfort of the passenger on an autonomous wheelchair, while guaranteeing the obstacle avoidance requirements. The study of comfort, which is a subjective feeling of the user, is treated objectively considering the vibrations of human body, which are proved to be related to discomfort in numerous studies, the main one being [2].

To reach this objective, the approach utilized has been to define a set of constraints restricting the control sequence based on the studies of human ride comfort, that are included into the MPC algorithm to be considered when calculating the optimal trajectory.

1.2 Structure of the thesis

The thesis is structured as follows:

- In Chapter 2, current studies of human comfort in autonomous navigation are presented, as well as the elements of the real-world system.
- Chapter 3 provides a general description of the control technique used, *MPC*, defining its formulation, its functioning and how to impose the conditions of stability and robustness.
- Chapter 4 presents the model of the system, as well as the methodology to get rid of its non-linearities in order to be able to use it in the control process.
- Chapter 5 contains a detailed explanation of the algorithm which, at every iteration step, detects and processes the obstacles near the wheelchair, allowing to define an admissible region for the robot position.
- Chapter 6 defines the concept of ride comfort and how to evaluate it objectively, with the objective of including comfort constraints into the optimization problem.
- In Chapter 7, finally all the previous considerations are taken into account, and the specific *MPC* problem for this work is completely defined.
- Chapter 8 shows the results of the simulations effectuated in order to verify the correct functioning of the algorithm.
- Lastly, in Chapter 9, the conclusions of the project are discussed, as well as future developments on the work done.

Chapter 2

State of art

This chapter describes various approaches to treat ride comfort in autonomous vehicles, to illustrate the differences between the methodology currently investigated and the one used in this work. Additionally, to facilitate the understanding of the system in which the work is based, the *hardware* utilized in previous study is briefly introduced.

2.1 Experimental devices

- **Wheelchair:** *Degonda Twist t4 2x2*, an electric wheelchair with two rear driven wheels with independent actuators with a power of $0.35kW$ each. Additionally it has three non-driven wheels, one in the back and two in the front, to act as stabilizers.



Figure 2.1: *Degonda Twist t4 2x2*

- **Computer:** compact computer *Shuttle DS81L*, designed specifically for robotics, utilizing *Linux* as its operating system.
- **Communication bus:** to transfer the data measured by the sensors and other components of the system to the computer.

- **Encoder:** the angular velocity of each wheel is measured utilizing two independent rotary encoders. Based on these values, it is possible to calculate the longitudinal and angular velocity of the wheelchair v and ω .
- **Laser sensors:** the wheelchair has two time-of-flight laser sensors *SICK TiM561*, which feature the following specifications:
 - Working range: $0.05\text{ m} - 10\text{ m}$
 - Aperture angle: 270°
 - Angular resolution: 0.33°
 - Scanning frequency: 15 Hz

These sensors are placed and integrated adequately to achieve a 360° scanning.

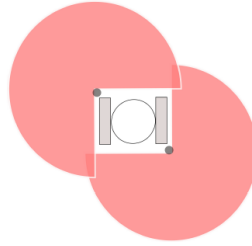


Figure 2.2: Position of the laser sensors, from [1]

2.2 Modelling of a comfort map based on subjective experiments

One approach to consider the comfort of the user in autonomous navigation is to build a comfort map defining which areas are perceived as more comfortable by the user, as seen in [3]. This map is built based on subjective experience gathered through experiments to later be integrated to a geometric map generated by a *SLAM* framework, which the global planner utilizes to generate a safe and comfortable path.

The comfort map defined consists of various areas, as seen in Figure 2.3:

- **Unsafe space**(blue): includes the *collision* region and the *vehicle over specification* region.
- **Safe space**(blue to yellow): this is the region where the vehicle is collision free and encompasses the comfort and discomfort spaces. In this space, navigation is safe but not comfortable.
- **Comfort region**(yellow): this is the space where the human passenger feels safe and comfortable.

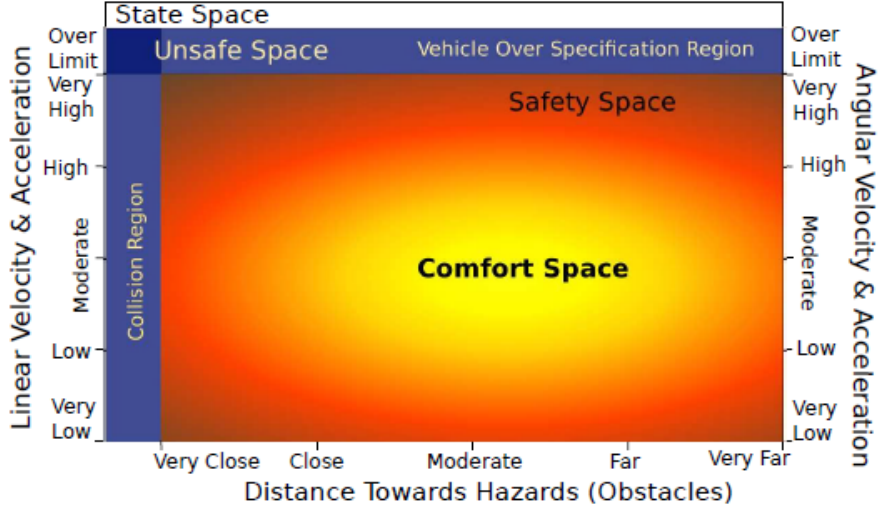


Figure 2.3: Comfort map definition, from [3]

The bottom left shows the discomfort of traveling very close to obstacles and the top left represents the *fear* of traveling close to obstacles at very high velocities. The region at the top right represents the discomfort of traveling at very high velocities and bottom right region is the discomfort as the vehicle travels at very low velocities even without hazards around it.

The factors considered to define this comfort map are:

- d_h : distance between the center of mass of the human to the closest object
- \dot{x} : linear velocity of the wheelchair
- $\dot{\theta}$: angular velocity of the wheelchair
- \ddot{x} : linear acceleration
- $\ddot{\theta}$: angular acceleration

The model presented describes navigational comfort in a straight corridor environment in terms of distance from the wall and linear velocity, using the energy expression

$$U(d_h, \dot{x}) = 1 - \left(\frac{c_a \dot{x}}{d_h} + \frac{(\dot{x} - V_0)^2}{c_b^2} + \frac{(d_h - k_0 L)^2}{c_c^2} \right) \quad (2.1)$$

where V_0 is the preferred velocity, $k_0 L$ is the preferred position on the corridor (being k_0 a percentage and L the width of the corridor), which are determined via interviews to participants of the experiments.

The model coefficients c_a , c_b and c_c are obtained through a regression analysis on the data and represent:

- c_a : trade off between velocity and distance

- c_b : weight of the velocity
- c_c : weight of the position within the corridor

This function presents a maximum at the highest grade of comfort. The energy function is then added to a geometric map, which is used later to find the most comfortable path using A^* algorithm.

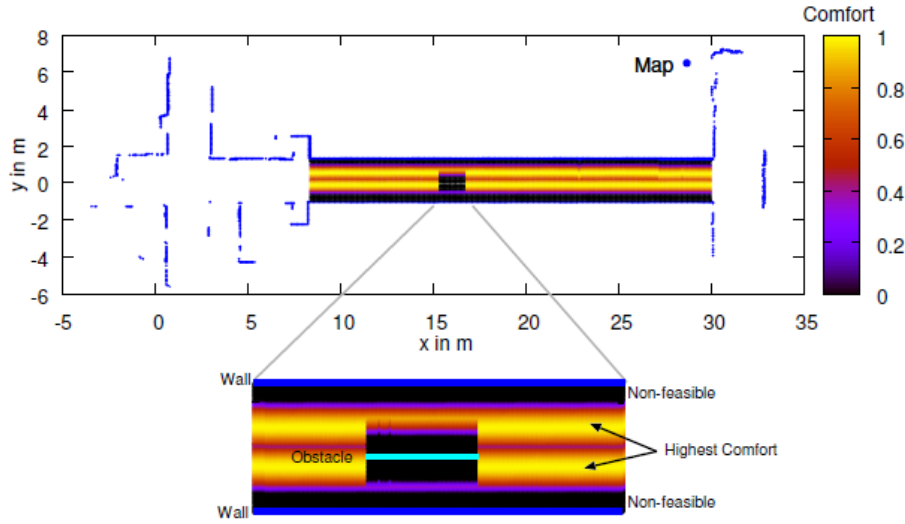


Figure 2.4: Comfort map of a corridor with an obstacle, from [3]

The cost function to be minimized by the A^* algorithm represents discomfort, and is expressed as

$$f(x) = k_D(g(x) + h(x)) + (1 - k_D) m_{disc}(x, \dot{x}) \quad (2.2)$$

where $g(x)$ is the distance of the starting node to current node x , $h(x)$ is the distance from node x to the goal and $m_{disc}(x) = 1 - U(dh(x), \dot{x})$ is the discomfort cost of taking the path through node x . Parameter $k_D = 0.5$ is a weighting coefficient due to distance and discomfort.

The results of this study have been proven successful, as the path generated using this method resulted statistically more comfortable to the vast majority of participants.

2.3 Suppression of discomfort vibrations

Another approach to treat ride comfort in autonomous navigation is presented in [4]. This approach, which is based in the same principle as the one utilized in this thesis, relates human comfort with the frequency of the vibrations that are produced during the navigation, specially those with the natural frequency of the wheelchair and human organ.

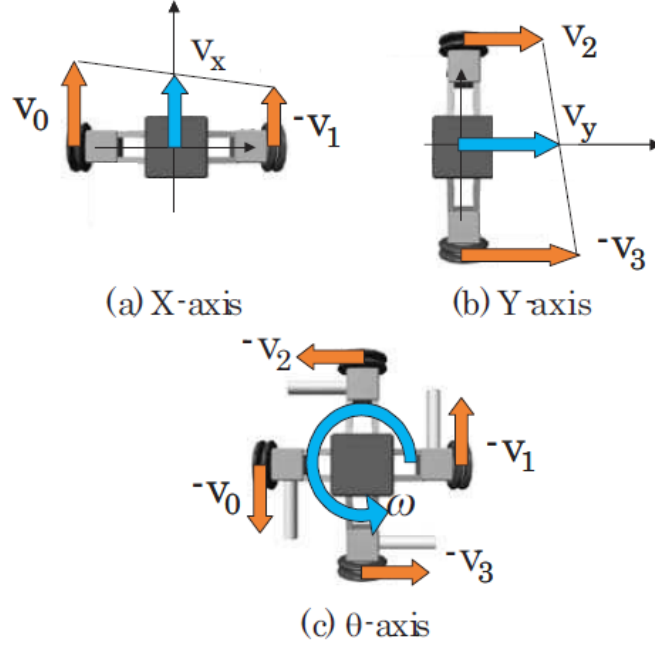


Figure 2.5: Axes of the OMW, from [4]

This study considers an Omni-directional Wheelchair (*OMW*), whose natural frequencies have been determined to be 2.4 Hz for the X direction and 2.45 Hz for the Y direction. On the other hand, the natural frequency of human's organ is in the range of $4 - 8 \text{ Hz}$, so it has been adopted as 6 Hz .

The objective is to apply a PI controller for the motion control, in series with a combination of filters to avoid exciting the system at the frequencies stated above.

The *PI* controller is chosen in order to compensate the steady state error of the servomotors. It is expressed as:

$$K_1(s) = K_P + K_I/s \quad (2.3)$$

Two notch filters are used to prevent the controller from exciting vibration of the *OMW* or user's organs.

$$K_2(s) = \frac{s^2 + 2\zeta_1\omega_1 s + \omega_1^2}{s^2 + \omega_1 s + \omega_1^2} \quad (2.4)$$

$$K_3(s) = \frac{s^2 + 2\zeta_2\omega_2 s + \omega_2^2}{s^2 + \omega_2 s + \omega_2^2} \quad (2.5)$$

where natural frequency of the *OMW*, $\omega_1 = 15.08 \text{ rad/s}$ (2.40 Hz), in case of Y axis $\omega_1 = 15.39 \text{ rad/s}$ (2.45 Hz); natural frequency of human's organs $\omega_2 = 37.7 \text{ rad/s}$ (6 Hz); damping ratio $\zeta_1 = \zeta_2 = 0.0001$

Furthermore, a low pass filter is also applied to reduce the influence of high-order vibration and noise:

$$K_4(s) = \frac{1}{T_n s + 1} \quad (2.6)$$

In order to determine users' sway, a human model was developed in [4]. This model considers the human upper body consisting of two rigid parts, head and torso.

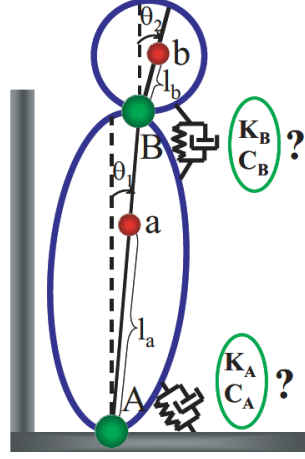


Figure 2.6: Human model, from [4]

User is considered to be supported on the wheelchair only at one point: point A , because the contact pressure is the strongest at this point. Point a and b are the center of gravity of torso and head, respectively. l_a is defined by the distance between point A and point a , and l_b is the distance between point B and point b .

The equation of motion using generalized coordinates can be expressed as:

$$\mathbf{M}\{\ddot{q}\} + \mathbf{C}\{\dot{q}\} + \mathbf{K}\{q\} + \mathbf{g}(q) = \{Q\} \quad (2.7)$$

where $q = \{\theta_1, \theta_2\}$ are the generalized coordinates and $Q = 0$ is the applied force and,

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad (2.8)$$

with

$$\begin{aligned} M_{11} &= m_1 l_a^2 + m_2 l_1^2 + J_1 \\ M_{12} &= M_{21} = m_2 l_1 l_b \cos(\theta_1 - \theta_2) \\ M_{22} &= m_2 l_b^2 + J_2 \end{aligned}$$

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \quad (2.9)$$

with

$$\begin{aligned} C_{11} &= C_A + C_B \\ C_{12} &= m_2 l_1 l_b \sin(\theta_1 - \theta_2) \dot{\theta}_2 - C_B \end{aligned}$$

$$\begin{aligned} C_{21} &= -m_2 l_1 l_b \sin(\theta_1 - \theta_2) \dot{\theta}_2 - C_B \\ C_{22} &= C_B \end{aligned}$$

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \quad (2.10)$$

with

$$\begin{aligned} K_{11} &= K_A + K_B \\ K_{12} &= -K_B \\ K_{21} &= -K_B \\ K_{22} &= K_B \end{aligned}$$

$$\mathbf{g} = \begin{bmatrix} g(\theta_1) \\ g(\theta_2) \end{bmatrix} \quad (2.11)$$

with

$$\begin{aligned} g(\theta_1) &= (m_2 l_1 \ddot{x} + m_1 l_a \ddot{x}) \cos \theta_1 \\ g(\theta_2) &= m_2 l_b \ddot{x} \cos \theta_2 \end{aligned}$$

Here, K_A , C_A , K_B and C_B , which are the spring and damping constants between torso and seat and between torso and head respectively, have been identified by means of motion capture.

Once the model is built, it is used to determine the vibration of the head of the user, which is found to be 8.17 rad/s (1.3 Hz). This frequency of vibration is again used to define a new notch filter which is added to the controller:

$$K_5(s) = \frac{s^2 + 2\zeta_3 \omega_3 s + \omega_3^2}{s^2 + \omega_3 s + \omega_3^2} \quad (2.12)$$

with $\zeta_3 = 0.001$ and $\omega_3 = 8.17 \text{ rad/s}$ (1.3 Hz).

Finally, the controller is given as

$$\begin{aligned} K(s) &= \prod_{i=1}^5 K_i(s) = \\ &= \frac{(K_P s + K_I)(s^2 + 2\zeta_1 \omega_1 s + \omega_1^2)(s^2 + 2\zeta_2 \omega_2 s + \omega_2^2)(s^2 + 2\zeta_3 \omega_3 s + \omega_3^2)}{s(s^2 + \omega_1 s + \omega_1^2)(s^2 + \omega_2 s + \omega_2^2)(T_n s + 1)(s^2 + \omega_3 s + \omega_3^2)} \end{aligned} \quad (2.13)$$

where K_P , K_I and T_n are unknown parameters found through optimization. In the formulation of this optimization, the gain at the frequencies of interest is set to be less than 0 dB:

$$\begin{cases} |K(\omega_1)| < 0dB \\ |K(\omega_2)| < 0dB \\ |K(\omega_3)| < 0dB \end{cases}$$

This results in a controller that allows to suppress the undesired vibrations and thus improving the comfort of the user.

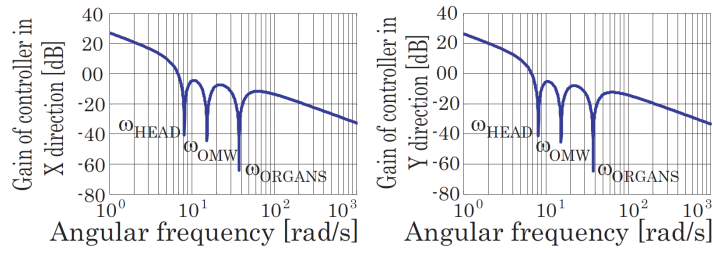


Figure 2.7: Controller to suppress undesired vibrations, from [4]

Chapter 3

Model predictive control

Model predictive control (MPC) is an advanced control method that is used to control a process while satisfying a set of constraints. This technique has been mainly used in the process industries in chemical plants and oil refineries since the 1980s. In recent years it has also been used in power system balancing models and in power electronics.

The main characteristic of this method is the possibility to control multi-variable systems considering the optimal evolution of the state variables during a control horizon, which is defined beforehand. To do so, the controller relies on dynamic models of the process, most often linear empirical models obtained by system identification, but also models obtained by analytical modeling of the physic system.

Another important characteristic is the ability to define constraints on the input signal and state variables, which guarantees the optimal solution to be found is adequate to the physical limitations of the process, providing robustness into the control law.

Thus, the objective of the controller is to compute the optimal signal input in the defined horizon, taking as initial point the previous state of the system. To achieve this, it is necessary to define a cost function, usually including the square sums of the error for both state variables and control signal. The objective, then, is to minimize the error via an optimization problem.

Another factor that allows a more robust solution is the utilization of the so-called *Receding Horizon Strategy*, which main idea is to compute, at every iteration, the optimal input sequence for the defined horizon, and applying only the first control signal of this result to the system. This process is repeated at each instant setting as a starting point the state on the previous instant. This idea can be seen in detail in Figure 3.1.

Assume a discrete-time setting and that the current time step is labelled as time step k . At the current time the system output is $y(k)$, and the figure shows the previous history of the output. Also shown in the figure is the *set-point trajectory*, which is the trajectory the output should follow ideally, which is named $s(t)$.

Different from the set-point trajectory there is the **reference trajectory**. This trajectory starts at the current output $y(k)$ and defines an ideal trajectory for the plant to return to the set-point trajectory, for instance after a disturbance occurs.

Using the *internal model*, the controller predicts the behaviour of the plant, starting at the current time, over a future prediction horizon H_p . This predicted

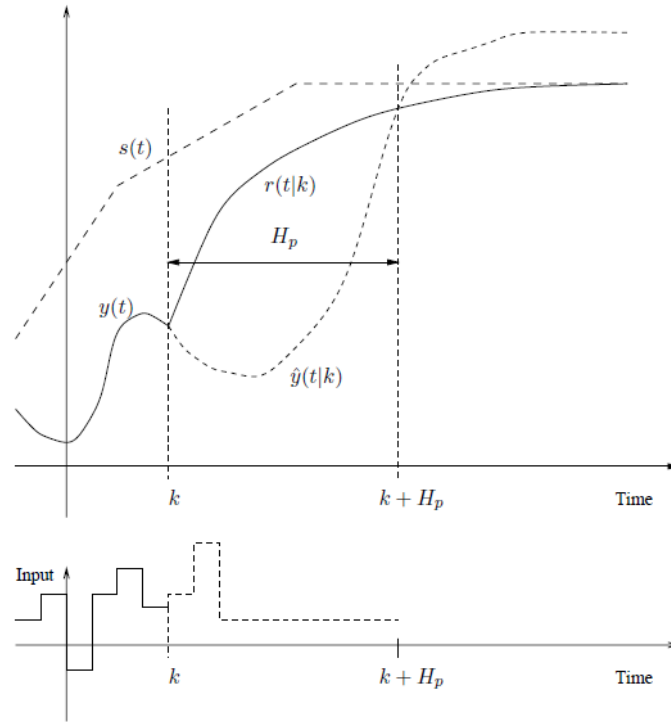


Figure 3.1: Receding horizon strategy

behaviour depends on the input trajectory $\hat{u}(k+i|k)$ ($i = 0, 1, \dots, H_p - 1$) that is to be applied over the prediction horizon, and the idea is to optimally select this input. The notation \hat{u} rather than u indicates that this input sequence is an estimate at the time instant k , but it may vary when reaching the instant $k+i$.

Once a future trajectory has been chosen, only the *first element* of that trajectory is applied as the input signal to the plant, that is $u(k) = \hat{u}(k|k)$. Then the whole cycle of output measurement, prediction and input trajectory determination is repeated, one sampling interval later. Since the prediction horizon remains of the same length as before, but every instant it starts one sampling step later, this way of controlling a plant is called a *receding horizon strategy*.

The main advantage of this methodology is that it guarantees a closed loop control, as the calculation of the optimal control signal at each iteration allows to avoid errors due to the uncertainty on input and output, as well as the error due to the imprecision introduced by the model.

3.1 System model

The general formulation used considers a linear, discrete-time, state model of the plant, with m input signals n state variables, and l outputs. This model is expressed in the form

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) \quad (3.1)$$

$$y(k) = \mathbf{C}x(k) \quad (3.2)$$

Where x is an n -sized vector containing the state variables, u is an m -sized vector containing the input signal and y an l -sized vector containing the output of the system. The output is assumed to be observable and coincident with the state variables.

The system is subject to a set of constraints applied to both the state variables and the output, which correspond to physical limitations of the system or control requirements. These sets of constraints can be expressed as

$$x(k) \in \mathbb{X} \subset \mathbb{R}^n, \quad u(k) \in \mathbb{U} \subset \mathbb{R}^m \quad (3.3)$$

where \mathbb{X} and \mathbb{U} are the admissible regions for the state variable and the input respectively, both containing the origin.

3.2 MPC basic formulation

As stated before, the main goal of MPC is to find, at each instant, the optimal input signal to be applied to the system so it reaches the reference state within the prediction horizon, while satisfying a set of constraints.

Given a prediction horizon N , the objective is to determine the vector $\mathbf{u}(k)$

$$\mathbf{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

containing the input sequence that minimizes the cost function

$$J(\mathbf{x}(k), \mathbf{u}(k)) = \sum_{i=0}^{N-1} f(\mathbf{x}(k+i), \mathbf{u}(k+i)) + V^f(\mathbf{x}(k+N)) \quad (3.4)$$

In equation (3.4), $f(\mathbf{x}(k+i), \mathbf{u}(k+i))$ is a positive-definite function of the state and input variables called *stage cost*, while $V^f(\mathbf{x}(k+N))$ is the so-called *terminal cost*, a function depending on the final state of the system.

For the formulation used in this thesis, it is assumed that the plant model is linear, that the cost function is quadratic, and that constraints are in the form of linear inequalities. It is also assumed that everything is time-invariant.

The sub-functions that compose the cost function are usually chosen as quadratic functions of the form

$$\begin{aligned} f(\hat{\mathbf{x}}(k+i|k), \mathbf{u}(k+i)) &= \|\hat{\mathbf{x}}(k+i|k)\|_{\mathbf{Q}}^2 + \|\mathbf{u}(k+i)\|_{\mathbf{R}}^2 \\ V^f(\hat{\mathbf{x}}(k+N|k)) &= \|\hat{\mathbf{x}}(k+N|k)\|_{\mathbf{P}}^2 \end{aligned}$$

where \mathbf{Q} , \mathbf{R} and \mathbf{P} are weight matrices, symmetric and real, associated to the error of the state variables, the input signal and the final state, respectively. \mathbf{Q} and \mathbf{P} are *positive semidefinite*, while \mathbf{R} is *positive definite*. The choice of these matrices is explained in Section 7.2.

Because the optimization problem to be solved depends only on the input sequence, it is necessary to transform the expressions of the cost function and constraints. To accomplish this, the relation between the state variables and the input signal is determined by the solution of (3.1), which is

$$\hat{\mathbf{x}}(k+i|k) = \mathbf{A}^i \mathbf{x}(k) + \sum_{j=0}^{i-1} \mathbf{A}^{i-j-1} \mathbf{B} \mathbf{u}(k+j) \quad (3.5)$$

3.3 Stability

The parameters of the control have to be carefully chosen to guarantee the stability of the closed-loop system. Specially, the *terminal cost* of the cost function has to be determined using an auxiliary control law, defined arbitrarily, to ensure the stability of the system. The auxiliary control law is expressed as

$$\mathbf{u}(k) = \mathbf{K}_a \mathbf{x}(k)$$

which, substituted in (3.1), leads to the system expression

$$\mathbf{x}(k+1) = (\mathbf{A} + \mathbf{B} \mathbf{K}_a) \mathbf{x}(k) \quad (3.6)$$

In this way, to ensure the stability of the system, it is necessary to choose \mathbf{K}_a adequately to make the matrix $(\mathbf{A} + \mathbf{B} \mathbf{K}_a)$ stable. Usually, this is made by assignation of the eigenvalues or using the gain of the LQ control. Once an adequate value of \mathbf{K}_a is defined, the matrix \mathbf{P} for the *terminal cost* can be calculated using the discrete *Lyapunov equation*

$$(\mathbf{A} + \mathbf{B} \mathbf{K}_a)' \mathbf{P} (\mathbf{A} + \mathbf{B} \mathbf{K}_a) - \mathbf{P} = (\mathbf{Q} + \mathbf{K}_a' \mathbf{R} \mathbf{K}_a) \quad (3.7)$$

3.4 Recursive feasibility

To assure the feasibility of the control problem over time, it is advisable to define a more restrictive terminal set for the final state, containing the initial state. The terminal set \mathbb{X}_t has to be completely contained in the admissible region for the state \mathbb{X} , as well as be characterized by the *positive invariance* property.

$$\hat{\mathbf{x}}(k+N|k) \in \mathbb{X}_t \subset \mathbb{X}$$

Given the auxiliary control law

$$\mathbf{u}(k) = \mathbf{K}_a \mathbf{x}(k)$$

the set \mathbb{X}_t is *positive invariant* respect the closed loop system (3.6) if

$$\mathbf{x}(\bar{k}) \in \mathbb{X}_t \Rightarrow \mathbf{x}(k) \in \mathbb{X}, \quad \forall k \geq \bar{k}$$

which leads to the following requirement regarding the control signal

$$\mathbf{u}(k) = \mathbf{K}_a \mathbf{x}(k) \subseteq \mathbb{U}, \quad \forall \mathbf{x}(k) \in \mathbb{X}_t$$

Thereby, setting an initial state such that $\mathbf{x}(\bar{k}) \in \mathbb{X}_t$ and applying an auxiliary control law asymptotically stable, the state will remain within the terminal set \mathbb{X}_t for every iteration, while satisfying the desired constraints.

3.5 Definition of the optimization problem

Taking into account the considerations presented previously, the control is described as follows

$$\begin{aligned} \min_{\mathbf{u}(k)} \quad & J(\mathbf{x}(k), \mathbf{u}(k)) \\ \hat{\mathbf{x}}(k+i|k) & \in \mathbb{X}, \quad \mathbf{u}(k+i) \in \mathbb{U} \quad \forall i \in \{0, \dots, N-1\} \\ \hat{\mathbf{x}}(k+N|k) & \in \mathbb{X}_t \end{aligned}$$

However, the definition of the control problem needs to be transformed so that it depends only on the input signal, in order to be able to introduce it in the optimization solver. The chosen formulation for the optimization problem consists of a quadratic cost function and linear constraints

$$\min_{\mathbf{u}(k)} \quad J(k) = \frac{1}{2} \mathbf{u}'(k) \mathbf{H} \mathbf{u}(k) + \mathbf{f}' \mathbf{u}(k) \quad (3.8)$$

$$s.t. \quad \mathbf{A}_{ineq} \mathbf{u}(k) \leq \mathbf{b}_{ineq}$$

Where \mathbf{u} is the solving variable, \mathbf{H} is the *Hessian matrix* (positive semidefinite) corresponding to the quadratic terms of the cost function and \mathbf{f} is the *gradient vector*, which corresponds to the linear terms.

\mathbf{A}_{ineq} and \mathbf{b}_{ineq} are a matrix and a vector, respectively, used to define the linear constraints to be applied to the optimization problem. The dimensions of these matrix depend on the number of constraints to be applied and the number of variables involved in the problem. The number of rows of both \mathbf{A}_{ineq} and \mathbf{b}_{ineq} correspond to the number of constraints, while the columns of \mathbf{A}_{ineq} match with the number of variables.

As stated before, the problem has to be rewritten in order to obtain a problem expressed only in function of the input signal, which can be done using the *Lagrange equation* (3.5):

$$\mathbf{x}(k) = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(k) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}\mathbf{B} & \mathbf{B} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}(k) \quad (3.9)$$

Thus, applying this relation to the cost function (3.4):

$$\begin{aligned} J(\mathbf{x}(k), \mathbf{u}(k)) &= \mathbf{x}'(k) \mathbf{Q} \mathbf{x}(k) + \mathbf{u}'(k) \mathbf{R} \mathbf{u}(k) = \\ &= (\mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k))' \mathbf{Q} (\mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k)) + \mathbf{u}'(k) \mathbf{R} \mathbf{u}(k) \end{aligned} \quad (3.10)$$

with

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \vdots \\ \mathbf{x}(k+N) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} \mathbf{u}(k) \\ \vdots \\ \mathbf{u}(k+N-1) \end{bmatrix}$$

$$\mathcal{Q} = \begin{bmatrix} Q & & & \\ & \ddots & & \\ & & Q & \\ & & & P \end{bmatrix} \quad \mathcal{R} = \begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}$$

After regrouping, the cost function can be written in as stated at (3.8):

$$J(k) = \frac{1}{2} \mathbf{u}'(k) \mathbf{H} \mathbf{u}(k) + \mathbf{f}' \mathbf{u}(k)$$

with the *Hessian* matrix and the *gradient* vector:

$$\mathbf{H} = \mathcal{B}' \mathcal{Q} \mathcal{B} + \mathcal{R}$$

$$\mathbf{f} = \mathcal{A}' \mathbf{x}'(k) \mathcal{Q} \mathcal{B}$$

Lastly, if it is necessary to apply constraints on the state variables, they can be transformed using the relation defined in (3.9). Given a set of constraints

$$\mathbf{A}_{state} \underbrace{(\mathcal{A} \mathbf{x}(k) + \mathcal{B} \mathbf{u}(k))}_{\mathbf{x}(k)} \leq \mathbf{b}_{state}$$

they can be rewritten as

$$\mathbf{A}_{ineq} \mathbf{u}(k) \leq \mathbf{b}_{ineq}$$

with

$$\mathbf{A}_{ineq} = \mathbf{A}_{state} \mathcal{B}$$

$$\mathbf{b}_{ineq} = \mathbf{b}_{state} - \mathbf{A}_{state} \mathcal{A} \mathbf{x}(k)$$

Chapter 4

System description

4.1 Kinematic model

The model used to describe the system is the so-called *Unicycle model*, which is suitable to represent a differential drive vehicle as the wheelchair. The kinematic model of an unicycle is:

$$\begin{cases} \dot{x}(t) = v(t) \cos \theta(t) \\ \dot{y}(t) = v(t) \sin \theta(t) \\ \dot{\theta}(t) = \omega(t) \end{cases} \quad (4.1)$$

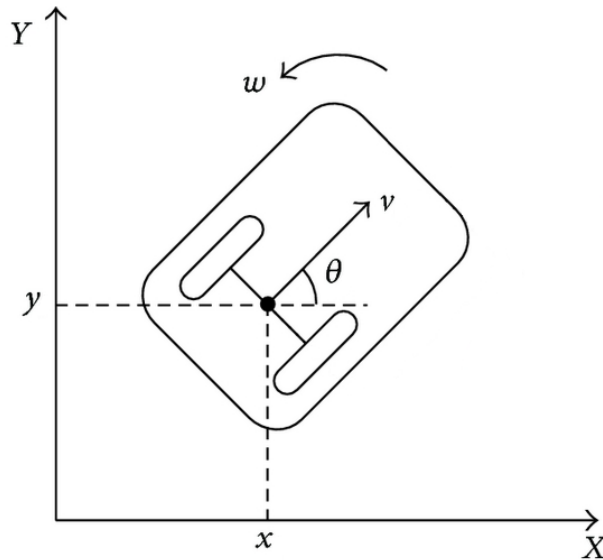


Figure 4.1: Unicycle model

In this model, the state variables are $x(t)$, $y(t)$ and $\theta(t)$, which represent the position of the center of the wheel axis expressed respect the global axes X and Y and the orientation of the vehicle, respectively. The control variables are $v(t)$ and $\omega(t)$ expressed again respect the intersection between the wheels.

However, in a differential drive unicycle, the control variables need to be expressed respect the speed of each wheel, as the actuators are usually electric motors attached to each wheel. The relation between these variables is:

$$\begin{cases} v(t) = \frac{v_L(t) + v_R(t)}{2} \\ \omega(t) = \frac{v_R(t) - v_L(t)}{L} \end{cases} \quad (4.2)$$

where L is the length of the axis (i.e., the distance between the wheels).

4.2 Feedback linearization

As thoroughly studied in [5], systems like the unicycle cannot be stabilized by any continuous and time-invariant control law. For this reason, it is common to study a point P outside the wheels' axis. The coordinates of this point P are given by:

$$P = \begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} x + \varepsilon \cos \theta(t) \\ y + \varepsilon \sin \theta(t) \end{bmatrix} \quad (4.3)$$

where ε is the longitudinal distance between the center of the axis and the point P , which is defined arbitrarily.

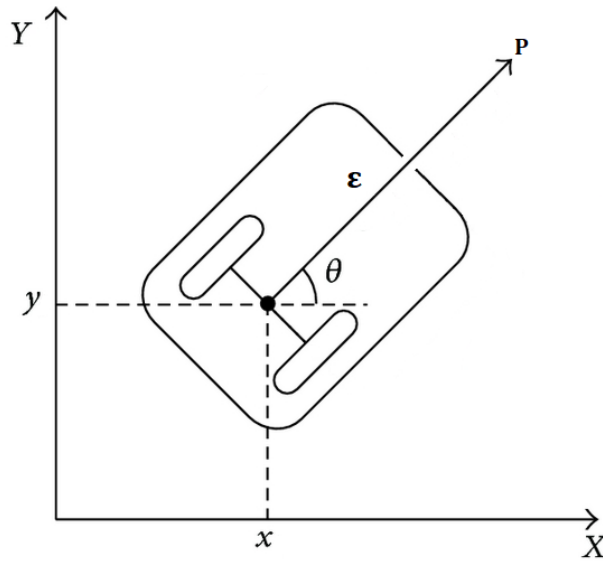


Figure 4.2: Study point, P

This change of point of study also allows to linearize the model, as the control variables of the differential drive unicycle ($v(t)$ and $\omega(t)$) present a non-linear relation with the state variables x and y . Thus it is possible to use a linearization method called *Feedback linearization*, consisting on the application of a linearizing closed loop to obtain a linear model, through a change of variables and the choice of an appropriate input signal.

Through the coordinates of the point P , stated at (4.3), the system can be described as:

$$\begin{cases} x_P(t) = x(t) + \varepsilon \cos \theta(t) \\ y_P(t) = y(t) + \varepsilon \sin \theta(t) \end{cases} \quad (4.4)$$

Deriving (4.4) with respect to time and substituting in the general expression of the unicycle model (4.1), we get to:

$$\begin{cases} \dot{x}_P(t) = v(t) \cos \theta(t) - \varepsilon \sin \theta(t) \omega(t) \\ \dot{y}_P(t) = v(t) \sin \theta(t) + \varepsilon \cos \theta(t) \omega(t) \end{cases} \quad (4.5)$$

which can be expressed in matrix form as

$$\begin{bmatrix} v_{Px}(t) \\ v_{Py}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}_P(t) \\ \dot{y}_P(t) \end{bmatrix} = \begin{bmatrix} \cos \theta(t) & -\varepsilon \sin \theta(t) \\ \sin \theta(t) & \varepsilon \cos \theta(t) \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \quad (4.6)$$

The feedback linearization matrix is, then:

$$T(\theta, \varepsilon) = \begin{bmatrix} \cos \theta(t) & -\varepsilon \sin \theta(t) \\ \sin \theta(t) & \varepsilon \cos \theta(t) \end{bmatrix} \quad (4.7)$$

which is not singular $\forall \theta$ and $\forall \varepsilon \neq 0$

Due to this non-singularity of the feedback linearization matrix, it can be inverted to find a relation between the input signal given by the controller, $v_{Px}(t)$ and $v_{Py}(t)$, and the original control signals of the unicycle model, $v(t)$ and $\omega(t)$. Thus, at each time instant, the orientation of the wheelchair can be determined easily:

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \cos \theta(t) & \sin \theta(t) \\ -\frac{1}{\varepsilon} \sin \theta(t) & \frac{1}{\varepsilon} \cos \theta(t) \end{bmatrix} \begin{bmatrix} v_{Px}(t) \\ v_{Py}(t) \end{bmatrix} \quad (4.8)$$

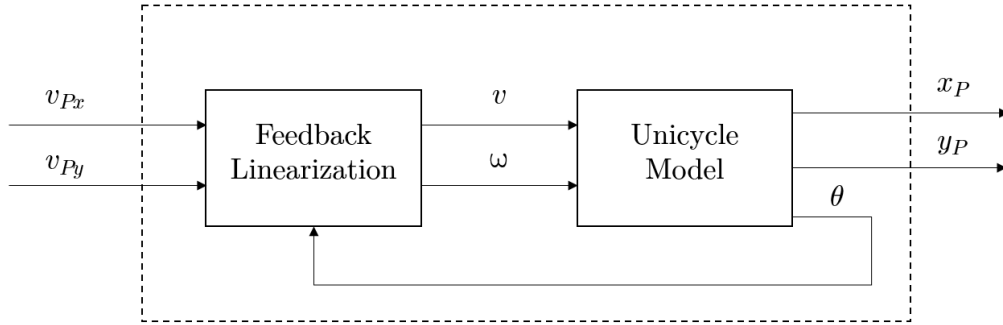


Figure 4.3: Control scheme with Feedback Linearization

The control diagram for using the feedback linearization can be seen at 4.3. The model obtained after the linearization is seen by the controller as a decoupled linear system consisting of two integrators, as the one showed in 4.4

$$\begin{cases} \dot{x}_P(t) = v_{Px}(t) \\ \dot{y}_P(t) = v_{Py}(t) \end{cases} \quad (4.9)$$

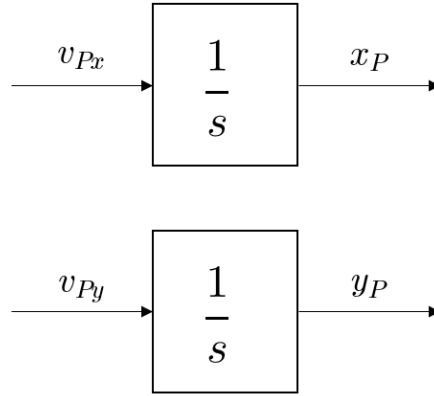


Figure 4.4: System seen by the controller

4.3 Discrete-time model

As the controller to be implemented is a digital controller, the model needs to be expressed as a discrete-time model with a sample time τ forced by the controller. The discrete model describing the system is

$$\begin{cases} x_P(k+1) = x_P(k) + \tau v_{Px}(k) \\ y_P(k+1) = y_P(k) + \tau v_{Py}(k) \end{cases} \quad (4.10)$$

which can be written in compact form as:

$$\mathbf{x}(k+1) = \mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k) \quad (4.11)$$

with

$$\mathbf{x}(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix}$$

Chapter 5

Obstacle avoidance

This chapter describes the process followed for the implementation of obstacle avoidance in the control algorithm. The main idea is to discard the trajectories that include collisions with the obstacles, which are detected at each sampling instant, to then compute the optimal trajectory solving the optimization problem defined at Chapter 3. To achieve this, the obstacles are included in the optimization problem after an adequate simplification.

However, this approach clashes with an important limitation of linear MPC control: the admissible region of the robot, which is defined as a set of constraints on the state variables, must be convex. Due to the morphology of the robot's environment, where the profile of the obstacles to be avoided include irregularities, the convexity of the admissible area detected is not guaranteed.

The solution adopted, then, is to design an algorithm which can compute a convex admissible region excluding the obstacles, while at the same time trying to maximize the work area of the robot, so as to not limit the control performance.

In this work, because a physical implementation has not been carried out, the point clouds have been generated computationally. These point clouds would be generated from the data gathered by a laser scan in the real wheelchair system.

5.1 Obstacle detection and segmentation

The starting data for the obstacle detection procedure is a point cloud provided by the laser sensor, which is an array of ordered points including the nearest obstacles respect the current position of the robot.

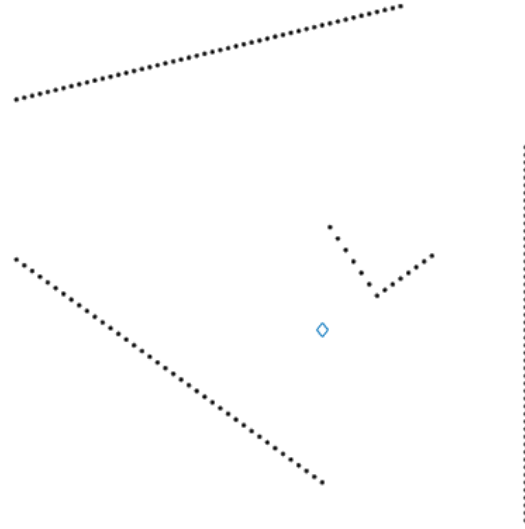


Figure 5.1: Point cloud with the detected points. The blue diamond represents the current position of the wheelchair.

The first step is to determine the number of physical obstacles defined by this set of points. To do so, it is necessary to go down the point cloud, checking that two consecutive points are nearer than an arbitrarily set distance threshold. If the threshold is exceeded, then the points are considered to belong to different physical obstacles. After checking the whole point cloud, the result is a new array of arrays, containing the list of points of each obstacle.

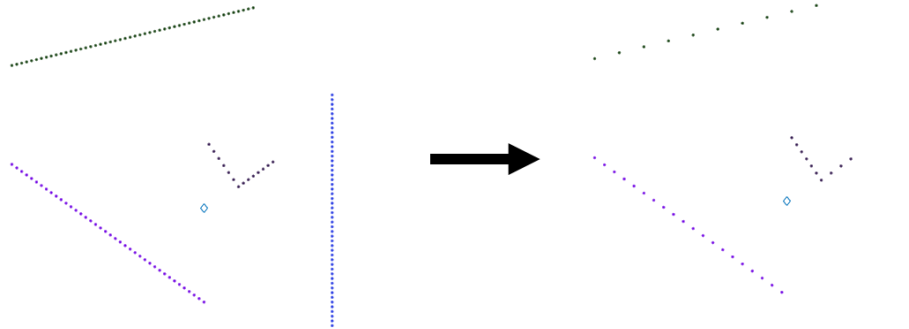


Figure 5.2: Divided obstacles and subsequent segmentation. The blue diamond represents the current position of the wheelchair.

Once the different obstacles are separated, the next step is to segment the objects in order to ease the computational load in later calculations. The segmentation algorithm employed to achieve this (which can be seen in [1]) guarantees the maintenance of the original shape of the obstacle while removing the points that are not necessary.

5.2 Constraints on state variables

As stated before, the data needed to apply the obstacle avoidance using MPC is a convex admissible space for the state variables. However, generally the point cloud obtained by the sensor is not convex, thus it is necessary to find a method to determine a convex admissible space as large as possible.

The usual solution to define this admissible space is to apply the so-called method of the *most violated constraint*, studied widely and successfully applied in [6]:

Definition 1. Starting on the basis that the equations which define the obstacle are known, it is possible to impose only one of these equations as a constraint according to the relative position of the vehicle, while ensuring the non-violation of the entire set of constraints. The choice of the linear constraint to be imposed is made considering the most distant constraint among those violated. i.e. those for which the current position of the vehicle and the obstacle belong to two different half-planes.

5.3 Convexification of the admissible region

The strategy chosen to impose the constraints for the obstacle avoidance, as explained previously, needs a convex admissible region for the state variables. Due to the earlier procedure to obtain the point cloud, it is usual that the obstacles detected are a combination of convex and concave objects.

In order to solve this issue, the obstacles are subdivided into virtual objects, all of them convex. This is achieved using an algorithm which evaluates the convexity of every vertex, subdividing the previous and subsequent set of points into two different obstacles. In the limit case, i.e. when two consecutive vertices are concave, the segment between them becomes the new obstacle.

The algorithm used for this subdivision is proposed in [1]. The algorithm recursively scans each obstacle, from the second to the second-to-last point, evaluating the positions of the current obstacle point and of the vehicle itself in relation with the line connecting the adjacent obstacle points, which leads to two cases:

- The current point and the vehicle belong to the same half-plane (Figure 5.3). In this case the vertex is **convex**, so the algorithm continues to evaluate the following point.
- The current point and the vehicle do not belong to the same half-plane (Figure 5.4). In this case the vertex is **concave**, so the obstacle is subdivided, creating a new obstacle with all the previous points, and the algorithm continues with the evaluation of the following points as a separate object.

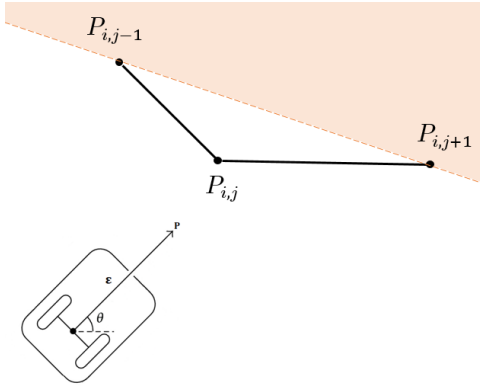


Figure 5.3: Convex set of points

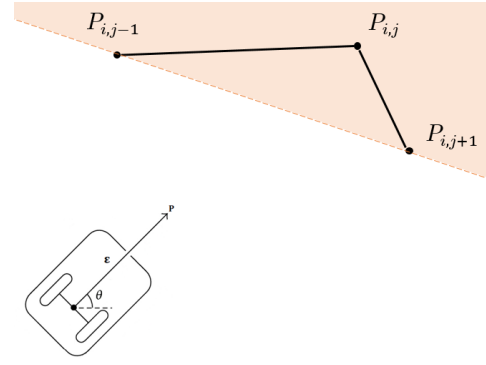


Figure 5.4: Concave set of points

Algorithm 1: *convex_politope*(*obstacles*, *pos*) :

```

concave = false;
for i = 0 to length(obstacles) - 1 do
  for j = 1 to length(obstacles[i]) - 2 do
     $P_{i,j} = \text{obstacles}[i][j];$ 
     $P_{i,j-1} = \text{obstacles}[i][j-1];$ 
     $P_{i,j+1} = \text{obstacles}[i][j+1];$ 
     $h_1 = P_{i,j-1}.y - P_{i,j+1}.y;$ 
     $h_2 = P_{i,j+1}.x - P_{i,j-1}.x;$ 
     $l = h_1 \cdot P_{i,j-1}.x + h_2 \cdot P_{i,j-1}.y$ 
    if  $(h_1 \cdot P_{i,j}.x + h_2 \cdot P_{i,j}.y - l) \cdot (h_1 \cdot \text{pos}.x + h_2 \cdot \text{pos}.y - l) < 0$  then
      concave = true;
      break;
    end
  end
  if concave then
    break;
  end
end
if concave then
   $n_{\text{obs}} = \text{length}(\text{obstacles});$ 
   $\text{obstacles}[n_{\text{obs}}] = \text{obstacles}[i][j : \text{end}];$ 
   $\text{obstacles}[i] = \text{obstacles}[i][0 : j];$ 
  convex_politope(obstacles, pos);
end

```

Once all the iterations through the list of obstacles are finished, the result is an array of arrays, called *obstacles*, containing all the virtual subdivisions of the obstacles and the points which compose them:

$$\text{obstacles} = \left[\underbrace{[P_{1,1} \ \dots \ P_{1,n}]}_{\text{obst}_1}, \ \dots, \ \underbrace{[P_{n_{\text{obs}},1} \ \dots \ P_{n_{\text{obs}},n}]}_{\text{obst}_{n_{\text{obs}}}} \right]$$

5.4 Definition of the position constraints

In order to accomplish the obstacle avoidance requirements, it is necessary to define an admissible region for the state variable. This can be achieved using the Plane Separation Axiom:

Definition 2. Euclidean half planes. Let $l = \overrightarrow{AB}$ be an Euclidean line. The Euclidean half planes determined by l are

$$H^+ = \{\mathbf{P} \in \mathbb{R}^2 \mid \langle \mathbf{P} - \mathbf{A}, (\mathbf{B} - \mathbf{A})^\perp \rangle > 0\}$$

$$H^- = \{\mathbf{P} \in \mathbb{R}^2 \mid \langle \mathbf{P} - \mathbf{A}, (\mathbf{B} - \mathbf{A})^\perp \rangle < 0\}$$

Notation. (\mathbf{X}^\perp or \mathbf{X}^{perp}). If $\mathbf{X} = (x, y) \in \mathbb{R}^2$, then $\mathbf{X}^\perp = (-y, x) \in \mathbb{R}^2$

This axiom can be used to define position constraints for the optimization problem, applying the belonging condition of the current position $\mathbf{P} = (x, y)$ for each obstacle. Considering two points from each obstacle ($\mathbf{A} = (x_a, y_a)$, $\mathbf{B} = (x_b, y_b)$), the inequation can be written as:

$$\begin{cases} (x - x_a) \cdot (y_a - y_b) + (y - y_a) \cdot (x_b - x_a) \geq 0 & \text{if } \mathbf{P} \in H^+ \\ (x - x_a) \cdot (y_a - y_b) + (y - y_a) \cdot (x_b - x_a) \leq 0 & \text{if } \mathbf{P} \in H^- \end{cases} \quad (5.1)$$

the direction of the inequality is defined based on the half-plane where the state needs to belong, depending on the relative position of the obstacle respect the vehicle.

To define this type of constraints more compactly and in order to obtain an expression which can be written in matrix form as a function of the state variables, the next form is adopted:

$$h_1 \cdot x + h_2 \cdot y \leq l \quad (5.2)$$

with

$$\begin{cases} h_1 = y_a - y_b \\ h_2 = x_b - x_a \\ l = h_1 \cdot x_a + h_2 \cdot y_a \end{cases}$$

Again, the signs of h_1 and h_2 need to be changed to impose the adequate semi-plane at the end of the process. Thus, applying a set of constraints containing one half-plane for every obstacle allows to define a convex admissible region for the state.

The first step consists in finding the pair of points which line is further from the current position of the vehicle, based on the method of the *most violated constraint*. It is important to note that, due to the segmentation process applied previously to achieve the convexity of the whole group of obstacles, every subset consists of at least two points.

At this point, theoretically it would be possible to just apply the line between the most distant points as a constraint, but this choice leads to a problem in the control process: segments which are sufficiently far and measured from the side are associated to lines very close to the current position of the vehicle, therefore the constraint calculated becomes too restricting.

In order to solve this problem, it is necessary to identify when this happens. As can be observed, the problem is presented when the projection of the position of

the wheelchair on the line associated to the segment is located outside the segment itself.

Geometrically, this corresponds to the situation where, given the current position P and the segment \overline{AB} , one of the angles \widehat{PAB} or \widehat{PBA} is obtuse, which can be determined using the dot product properties:

$$\overrightarrow{AP} \cdot \overrightarrow{AB} \leq 0 \quad \vee \quad \overrightarrow{BP} \cdot \overrightarrow{BA} \leq 0$$

Considering this situation, an adequate solution is to impose as a constraint the segment itself only in the case where the projection of the position is located inside the segment. Otherwise, only the closest point is considered (respect the current position), imposing as a constraint the perpendicular of the line connecting the vehicle and this point.

Given the current position P and the further couple of points of an obstacle in $obstacles[i]$, $O_{i,j}$ and $O_{i,j+1}$, there are two possible scenarios:

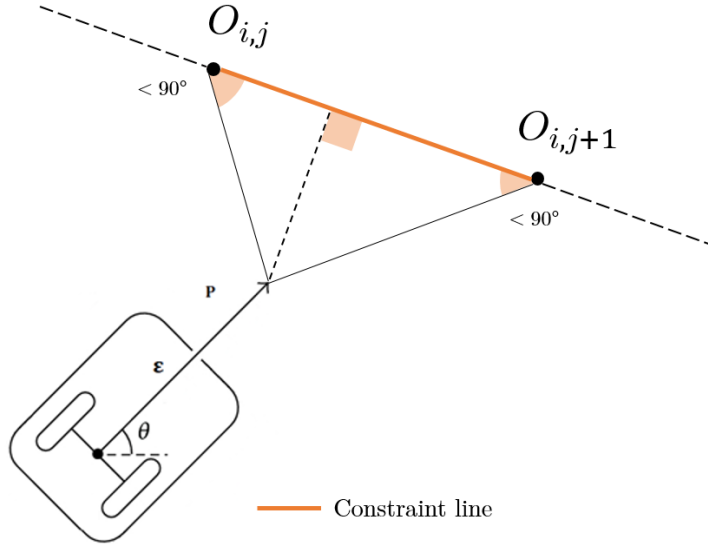


Figure 5.5: Internal projection of the current position

- **Internal projection** (Figure 5.5)

$$\text{Condition: } \overrightarrow{O_{i,j}P} \cdot \overrightarrow{O_{i,j}O_{i,j+1}} \geq 0 \quad \wedge \quad \overrightarrow{O_{i,j+1}P} \cdot \overrightarrow{O_{i,j+1}O_{i,j}} \geq 0$$

$$\text{Constraint line: } \begin{cases} h_1 = O_{i,j}.y - O_{i,j+1}.y \\ h_2 = O_{i,j+1}.x - O_{i,j}.x \\ l = O_{i,j+1}.x \cdot O_{i,j}.y - O_{i,j}.x \cdot O_{i,j+1}.y \end{cases}$$

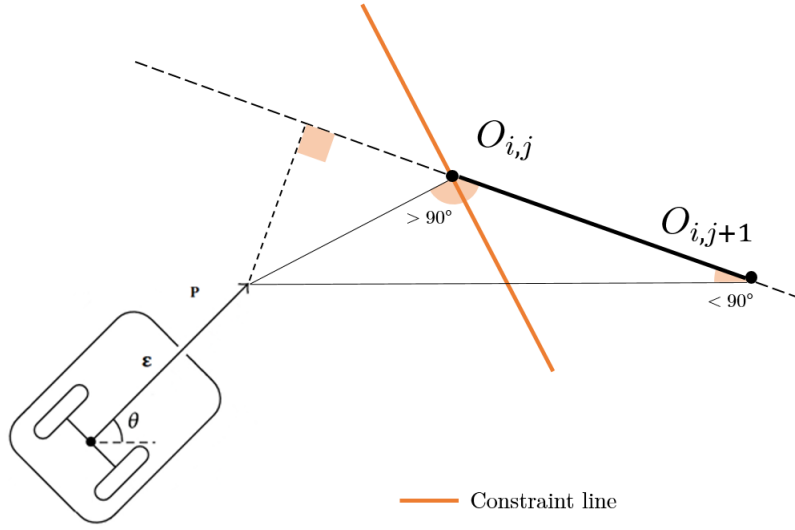


Figure 5.6: External projection of the current position

- **External projection** (Figure 5.6)

$$\text{Condition: } \overrightarrow{O_{i,j}P} \cdot \overrightarrow{O_{i,j}O_{i,j+1}} \leq 0 \quad \vee \quad \overrightarrow{O_{i,j+1}P} \cdot \overrightarrow{O_{i,j+1}O_{i,j}} \leq 0$$

$$\text{Closest point: } O_{min} = O_{i,k} \quad s.t. \quad \min_{k \in \{j,j+1\}} d(P, O_{i,k})$$

$$\text{Constraint line: } \begin{cases} h_1 = O_{min}.x - P.x \\ h_2 = O_{min}.y - P.y \\ l = h_1 \cdot O_{min}.x + h_2 \cdot O_{min}.y \end{cases}$$

Finally, the last step before writing the constraints is to determine the sign of the parameters h_1 , h_2 and l depending on which half-plane needs to be included into the admissible region. Due to the notation used throughout the process, the default parameters correspond to the H^+ semi-plane. Thus, a simple way of determining if the sign needs to be corrected is to check, for every object in $obstacles[i]$, if the current position belongs to this half-plane:

$$h_1^{(i)} \cdot x + h_2^{(i)} \cdot y \leq l^{(i)}$$

If the inequality is not satisfied, then it is necessary to change the sign of all the parameters, as the inequalities accepted by the optimization algorithm must be of the form $A x \leq b$. Thus, in this case the inequation to be imposed is:

$$\left(-h_1^{(i)}\right) \cdot x + \left(-h_2^{(i)}\right) \cdot y \leq \left(-l^{(i)}\right)$$

At this point, the constraints can be expressed in matrix form as:

$$\mathbf{A}_{state} \mathbf{x}(k) \leq \mathbf{b}_{state} \tag{5.3}$$

with

$$\mathbf{A}_{state} = \begin{bmatrix} h_1^{(1)} & h_2^{(1)} \\ \vdots & \vdots \\ h_1^{(nobs)} & h_2^{(nobs)} \end{bmatrix} \quad \mathbf{b}_{state} = \begin{bmatrix} l^{(1)} \\ \vdots \\ l^{(nobs)} \end{bmatrix}$$

Chapter 6

Ride comfort

This chapter describes the methodology employed to include the ride comfort concept into the control problem, in order to improve users' experience when using the vehicle.

Ride comfort is defined as the overall comfort and well-being of the vehicle's occupants during vehicle travel. It is one of the main factors to consider regarding the user, and while it is subjective and depends on the sensations of the user, it can be treated objectively. The main sources of discomfort are oscillations produced by external sources, so an usual approach is to study the response of the human body to the different vibrations produced during the ride of the vehicle.

The process of evaluating these vibrations has been standardized using the measure of acceleration at different points of the seat, the main examples being ISO 2631-1 [7] and BS 6841 [8], which treat the measure of whole body vibrations. In the case of study, ISO 2631-1 has been used as the reference standard.

6.1 Measurement and evaluation of vibration

As explained previously, the primary value used to measure the vibrations is acceleration. This acceleration needs to be measured according to a coordinate system originating at a point from which vibration is considered to enter the human body. In the case of seated position, there are three principal areas to be studied (Figure 6.1):

- **Seat-back** (1): a_x, a_y, a_z
- **Supporting seat surface** (2): $a_x, a_y, a_z, r_x, r_y, r_z$
- **Feet** (3): a_x, a_y, a_z

Considering that a_x, a_y and a_z are the linear accelerations at each axis, and r_x, r_y and r_z are the roll, pitch and yaw accelerations, respectively.

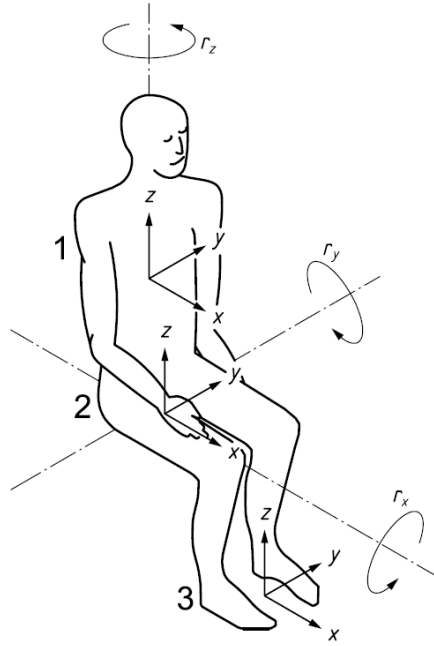


Figure 6.1: Basicentric axes of the human body, seated position

However, some researches note that the hip vibration show the highest correlation between objective and subjective data [9], so in the case of the wheelchair, it is possible to only consider the acceleration of the seat surface. This situation is really convenient, as the optimization problem is dependant on the speed of the feedback linearization point P (Section 4.2), so the acceleration can be expressed as a function of speed and be included directly into the optimization problem.

To evaluate the degree of vibration and consequently the ride comfort, it is usual to compute the CRV (component ride value) and the ORV (overall ride value), defined in [2].

The component ride value is defined as the effective magnitude of vibration at a single point (contact point between the human and the seat) after it has been weighted for frequency and axis according to human sensitivity.

$$CRV = \left[\frac{1}{N} \sum_{i=1}^N a_w^2(i) \right]^{1/2} \quad (6.1)$$

where $a_w(i)$ is the weighted acceleration and N is the number of acceleration signals at the point of study.

The overall ride value is defined as the effective magnitude of vibration occurring in one or more axes and input positions after it has been weighted for frequency, axis and input position according to human sensitivity. The overall ride value is evaluated as the 2-norm of the component ride value

$$ORV = \left[\sum CRV^2 \right]^{1/2} \quad (6.2)$$

As stated before, in the case of this project, only the linear accelerations at the

seat surface are considered, so there is only one value of CRV , and thus the ORV is the same as the ride value of the seat. Specifically:

$$ORV = CRV = \left[\frac{a_{w,Px}^2 + a_{w,Py}^2}{2} \right]^{1/2} \quad (6.3)$$

6.2 Frequency weighting of the acceleration

So as to obtain the value of ride comfort for the vehicle, it is necessary to previously filter every acceleration. ISO 2631 and BS 6841 introduce a set of weighting filters which are used to find a numerical equivalent to the comfort experienced by the user. In order to minimize the number of frequency weighting functions, some of them are combined for different axes, using *axis multiplying factors* k . Therefore, there exist 6 different weighting functions related to comfort:

- **Principal weighting functions:** W_k , W_d , W_f
- **Additional weighting functions:** W_c , W_e , W_j .

The weighting functions and *axis multiplying factors* for every area are described in Figures 6.2, 6.3 and Table 6.1, respectively.

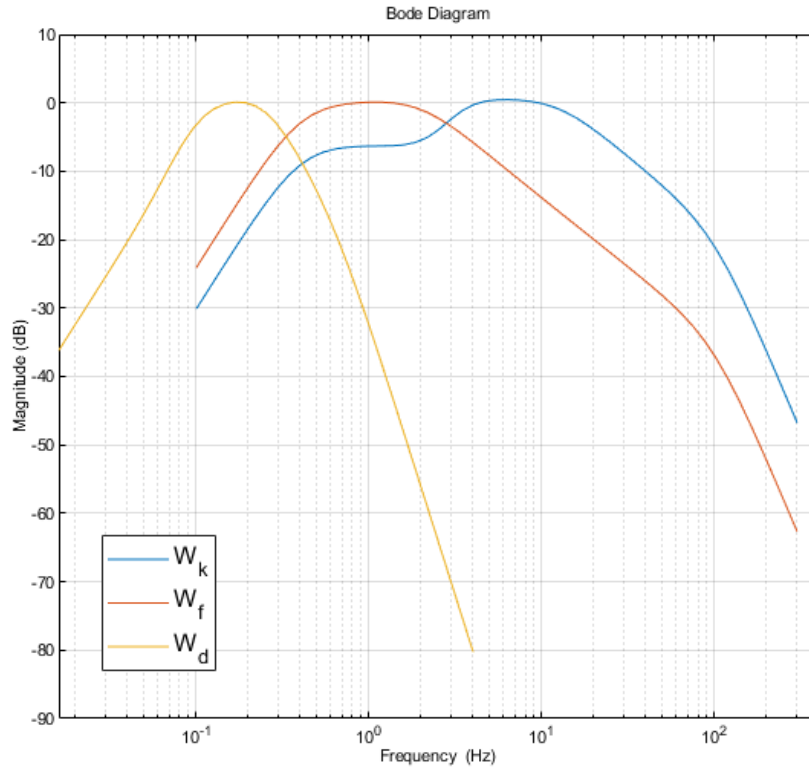


Figure 6.2: Principal frequency weighting functions, as described in ISO 2631-1

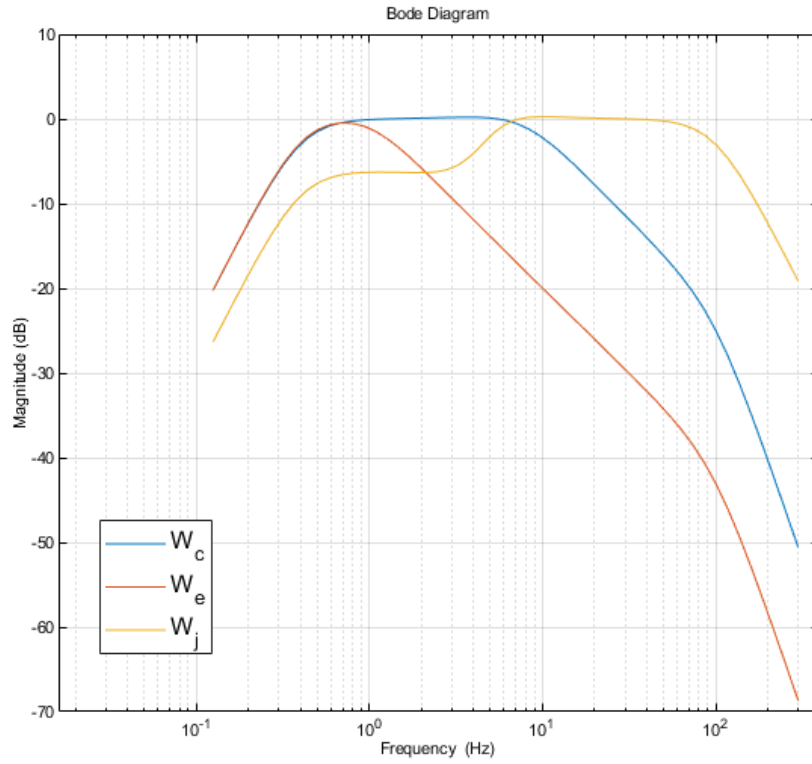


Figure 6.3: Additional frequency weighting functions, as described in ISO 2631-1

Area	Acceleration	Weighting function	Axis multiplying factor
Seat surface	a_x	W_d	$k = 1$
	a_y	W_d	$k = 1$
	a_z	W_k	$k = 1$
	r_x	W_e	$k = 0.63$
	r_y	W_e	$k = 0.4$
	r_z	W_e	$k = 0.2$
Backrest	a_x	W_c	$k = 0.8$
	a_y	W_d	$k = 0.5$
	a_x	W_d	$k = 0.4$
Feet	a_x	W_k	$k = 0.25$
	a_y	W_k	$k = 0.25$
	a_z	W_k	$k = 0.4$

Table 6.1: Weight function to be used in each area for seated persons, according to ISO 2631-1

To obtain the analytical expression of each filter (referred to acceleration as the input), it is necessary to combine a series of transfer functions, varying the parameters according to Tables 6.2 and 6.3. These transfer functions, expressed in the Laplace space, are:

- **Band-limiting** (two-pole filter with Butterworth characteristic, $Q_1 = Q_2 = 1/\sqrt{2}$):

– **High-pass**

$$H_h(s) = \frac{1}{1 + \sqrt{2} \omega_1/s + (\omega_1/p)^2} \quad (6.4)$$

where

$$\omega_1 = 2\pi f_1$$

f_1 = corner frequency (intersection of asymptotes)

– **Low-pass**

$$H_l(s) = \frac{1}{1 + \sqrt{2} s/\omega_2 + (s/\omega_2)^2} \quad (6.5)$$

where

$$\omega_2 = 2\pi f_2$$

f_2 = corner frequency

- **Acceleration-velocity transition** (proportionality to acceleration at lower frequencies, proportionality to velocity at higher frequencies)

$$H_t(s) = \frac{1 + s/\omega_3}{1 + s/(Q_4\omega_4) + (s/\omega_4)^2} \quad (6.6)$$

where

$$\omega_3 = 2\pi f_3$$

$$\omega_4 = 2\pi f_4$$

- **Upward step** (steepness of approximately 6dB per octave, proportionality to jerk):

$$H_s(s) = \frac{1 + s/(Q_5\omega_5) + (s/\omega_5)^2}{1 + s/(Q_5\omega_5) + (s/\omega_5)^2} \cdot \left(\frac{\omega_5}{\omega_6}\right)^2 \quad (6.7)$$

where

$$\omega_5 = 2\pi f_5$$

$$\omega_6 = 2\pi f_6$$

The product $H_h(s) \cdot H_l(s)$ represents the band-limiting transfer function, while the product $H_t(s) \cdot H_s(s)$ represents the actual weighting function for a certain application. The total weighting function is computed as:

$$H(s) = H_h(s) \cdot H_l(s) \cdot H_t(s) \cdot H_s(s) \quad (6.8)$$

A few particularities are that $H_t(s) = 1$ for weighting W_j and $H_s(s) = 1$ for weightings W_c , W_d and W_e , which is indicated by infinite frequencies and absence of quality factors Q in Tables 6.2 and 6.3.

Weighting	Band-limiting		a-v transition			Upward step			
	$f_1(Hz)$	$f_2(Hz)$	$f_3(Hz)$	$f_4(Hz)$	Q_4	$f_5(Hz)$	Q_5	$f_6(Hz)$	Q_5
W_k	0.4	100	12.5	12.5	0.63	2.37	0.91	3.35	0.91
W_d	0.4	100	2.0	2.0	0.63	∞	—	∞	—
W_f	0.08	0.63	∞	0.25	0.86	0.0625	0.80	0.1	0.80

Table 6.2: Parameters of transfer functions of the principal frequency weightings

Weighting	Band-limiting		a-v transition			Upward step			
	$f_1(Hz)$	$f_2(Hz)$	$f_3(Hz)$	$f_4(Hz)$	Q_4	$f_5(Hz)$	Q_5	$f_6(Hz)$	Q_5
W_c	0.4	100	8.0	8.0	0.63	∞	—	∞	—
W_e	0.4	100	1.0	1.0	0.63	∞	—	∞	—
W_j	0.4	100	∞	∞	—	3.75	0.91	5.32	0.91

Table 6.3: Parameters of transfer functions of the additional frequency weightings

In the case of the wheelchair, as stated before, the accelerations considered are only a_x and a_y of the seat surface, which according to the tables are filtered using the same weight function W_d with axis multiplication factor $k = 1$. Therefore, the transfer function defined to weight the acceleration is:

$$\frac{A_w(s)}{A(s)} = H(s) = k \cdot H_h(s) \cdot H_l(s) \cdot H_t(s) \quad (6.9)$$

6.3 Discretization and implementation of the weighting filter

The expression of the weighting filters provided by ISO 2631-1 corresponds to a continuous time transfer function, in the case of (6.9):

$$H(s) = \frac{0.08 s^3 + s^2}{1.6 \cdot 10^{-6} s^6 + 1.46 \cdot 10^{-5} s^5 + 0.007 s^4 + 0.15 s^3 + 1.50 s^2 + 4.37 s + 6.32}$$

However, this project is based on a digital controller, thus it is necessary to obtain a discrete-time expression in order to include the filtered acceleration into the control problem. The first step is to express the filter in the Z space using the Z transform, which results on a transfer function of the form:

$$H(z) = \frac{c_5 \cdot z^5 + c_4 \cdot z^4 + \dots + c_1 \cdot z + c_0}{b_6 \cdot z^6 + b_5 \cdot z^5 + \dots + b_1 \cdot z + b_0} \quad (6.10)$$

the coefficients $c_5, \dots, c_0 \in \mathbb{R}$ and $b_6, \dots, b_0 \in \mathbb{R}$ depend on the sampling time τ .

The transfer function (6.10) can also be expressed as

$$\frac{A_w(z)}{A(z)} = H(z) = \frac{c_5 \cdot z^{-1} + c_4 \cdot z^{-2} + \dots + c_1 \cdot z^{-5} + c_0 \cdot z^{-6}}{b_6 + b_5 \cdot z^{-1} + \dots + b_1 \cdot z^{-5} + b_0 \cdot z^{-6}}$$

Considering that the inverse Z transform of a term z^{-n} corresponds to a delay of n steps

$$\begin{aligned} Z^{-1}\{z^{-n} \cdot A_w(z)\} &= a_w(k - n) \\ Z^{-1}\{z^{-n} \cdot A(z)\} &= a(k - n) \end{aligned}$$

a discrete-time expression can be found for the weighted acceleration, which depends on the values of past filtered and linear accelerations:

$$a_w(k) = \frac{-b_5 \cdot a_w(k-1) - \dots - b_0 \cdot a_w(k-6) + c_6 \cdot a(k-1) + \dots + c_0 \cdot a(k-6)}{b_6} \quad (6.11)$$

With this expression it is now possible to implement constraints on the filtered acceleration as described in Section 6.1. The constraints implemented are:

$$\begin{aligned} ORV(k+i) &= \frac{1}{\sqrt{2}} \sqrt{a_{w,Px}(k+i)^2 + a_{w,Py}(k+i)^2} \leq ORV_{max} \quad (6.12) \\ \forall i &\in \{0, 1, \dots, N-1\} \end{aligned}$$

The value of ORV_{max} is defined based on the reference values for public transport provided by ISO 2631-1, which can be seen in Table 6.4. In the case of a real-world implementation, it would be necessary to find an adequate value experimentally.

$ORV[m/s^2]$	User's sensation
< 0.315	not uncomfortable
$0.315 - 0.63$	a little uncomfortable
$0.5 - 1$	fairly uncomfortable
$0.8 - 1.6$	uncomfortable
$1.25 - 2.5$	very uncomfortable
> 2	extremely uncomfortable

Table 6.4: Reference values of ORV for public transport

Chapter 7

MPC application

This chapter describes in detail how to define the control problem based on MPC to achieve an automatic navigation of the wheelchair in an environment partially known, which contains a series of obstacles that must be avoided while assuring the ride comfort of the wheelchair user.

Particularly, the chapter is focused on the formulation of the optimization problem to be solved at each sampling instant, which as explained in Chapter 3, allows to determine the optimal input sequence to get the vehicle to the reference point.

In order to be able to apply a linear control strategy, the *feedback linearization* method has been used, which allows to represent the unicycle model as a linear system. Using this technique, after an adequate change of variables, it is possible to obtain a linear model describing the behaviour of a point \mathbf{P} outside the wheels' axis, which is located at a distance ε from it. The state variables, then, become the position of this point \mathbf{P} , while the control variable is the speed of this point in each axis of the global reference X and Y . The detailed methodology can be seen at Section (4.2).

The *feedback linearization* technique allows, so, to compute the real values of the state variables v and ω at each sampling time, in function of the output generated by the controller.

The discrete model used to describe the unicycle, as explained previously, consists of two decoupled integrators, one for each axis. This system is defined as:

$$\mathbf{x}(k+1) = \mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k)$$

with

$$\mathbf{x}(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} v_{Px}(k) \\ v_{Py}(k) \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \tau & 0 \\ 0 & \tau \end{bmatrix}$$

The aim of this chapter is, then, to illustrate the choices made to implement the MPC control of the wheelchair system. As explained in Chapter 3, the main elements needed to achieve this are the *cost function*, the constraints to be imposed on the state variables and the input, and the *terminal set* for the state variables.

However, there are some particularities to consider regarding the wheelchair system:

- Constraints on the comfort ride value:
The necessity of filtering the acceleration to determine the ride comfort of the passenger leads to expressions of the acceleration that depend on previous instants, which combined to the non-linear nature of the constraints to be imposed, implies that these present non-linear relations between the control variables v_{Px} and v_{Py} . In order to be able to define an adequate set of constraints, it is necessary to linearize the constraints near the working point.
- Definition of the *terminal set* and the *terminal cost*:
It should be noted that the main use of optimal control in the industrial field is to achieve a robust regulation of processes that are linearized around time-invariant or slowly changing working points. Due to that, it is usual to constraint the control to converge towards the reference within the prediction horizon imposed. Thus, the choice of a terminal set is reduced to set an appropriate neighbourhood near the reference point. This practice is unworkable on a problem of autonomous navigation, as the constraints to be applied can result into the MPC optimization problem being unfeasible. For this reason the terminal set will be defined in a conservative manner using the same admissible region as the previous iteration and, specially, the state in the previous iteration is allowed to not necessarily be near the reference.

7.1 Cost function

As explained in Section 3.2, the control problem is defined as an optimization problem in a finite horizon N . The choice for this project is a quadratic cost function dependent on the error of the state and control variables towards a previously set reference, which is minimized so that this error tends to zero.

The reference for the state variables is defined as:

$$\mathbf{x}_{ref} = \begin{bmatrix} P_{x,ref} \\ P_{y,ref} \end{bmatrix}$$

Regarding the control variable, as the objective is to reach the position reference and make the vehicle remain there, the control signal should converge to zero

$$\mathbf{u}_{ref} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

As \mathbf{u}_{ref} is zero, it will be omitted in further formulation.

The cost function, as stated before, has a quadratic form and depends on the error of the variables towards their respective references:

$$J(k) = \sum_{i=0}^N (\|\mathbf{x}(k+i) - \mathbf{x}_{ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u}(k+i)\|_{\mathbf{R}}^2) + \|\mathbf{x}(k+N) - \mathbf{x}_{ref}\|_{\mathbf{P}}^2 \quad (7.1)$$

To write the optimization problem in the quadratic form presented in (3.8)

$$\min_{\mathbf{u}(k)} J(k) = \frac{1}{2} \mathbf{u}'(k) \mathbf{H} \mathbf{u}(k) + \mathbf{f}' \mathbf{u}(k) \quad (7.2)$$

it is necessary to write the cost function $J(k)$ in (7.1) in the compact form

$$\begin{aligned} J(k) &= \|\mathbf{x}(k) - \mathbf{x}_{ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u}(k)\|_{\mathbf{R}}^2 = \\ &= (\mathbf{x}(k) - \mathbf{x}_{ref})' \mathbf{Q} (\mathbf{x}(k) - \mathbf{x}_{ref}) + \mathbf{u}'(k) \mathbf{R} \mathbf{u}(k) \end{aligned} \quad (7.3)$$

with

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \vdots \\ \mathbf{x}(k+N) \end{bmatrix} \quad \mathbf{x}_{ref} = \begin{bmatrix} \mathbf{x}_{ref} \\ \vdots \\ \mathbf{x}_{ref} \end{bmatrix} \quad \mathbf{u}(k) = \begin{bmatrix} \mathbf{u}(k) \\ \vdots \\ \mathbf{u}(k+N-1) \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} Q & & & \\ & \ddots & & \\ & & Q & \\ & & & P \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}$$

Additionally, a further transformation is required in order to write $J(k)$ in function of the initial state and the control signal. This can be achieved through *Lagrange's equation* as explained in previous chapters (3.9).

$$\mathbf{x}(k) = \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(k) + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A} \mathbf{B} & \mathbf{B} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1} \mathbf{B} & \mathbf{A}^{N-2} \mathbf{B} & \dots & \mathbf{B} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}(k) \quad (7.4)$$

Therefore, the cost function becomes:

$$J(k) = (\mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k) - \mathbf{x}_{ref})' \mathbf{Q} (\mathbf{A} \mathbf{x}(k) + \mathbf{B} \mathbf{u}(k) - \mathbf{x}_{ref}) + \mathbf{u}'(k) \mathbf{R} \mathbf{u}(k)$$

This expression can be simplified to be written in the quadratic form, compatible with the optimization solver:

$$J(k) = \frac{1}{2} \mathbf{u}'(k) \mathbf{H} \mathbf{u}(k) + \mathbf{f}' \mathbf{u}(k) \quad (7.5)$$

with

$$\begin{aligned} \mathbf{H} &= \mathbf{B}' \mathbf{Q} \mathbf{B} + \mathbf{R} \\ \mathbf{f} &= (\mathbf{A} \mathbf{x}(k) - \mathbf{x}_{ref})' \mathbf{Q} \mathbf{B} \end{aligned}$$

7.2 Calibration of the controller

The cost function defined in the section above can be calibrated based on the requirements of the control. Particularly, the matrices \mathbf{Q} , \mathbf{R} and \mathbf{P} represent the penalization of the error of the state variables, the control signal and the final state, respectively.

The decoupling of the system into two integrators (one for each axis) and the symmetry of the model allows to impose a single parameter for each matrix, which will affect the variables of different axis equally:

$$\mathbf{Q} = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p & 0 \\ 0 & p \end{bmatrix}$$

The parameters q and r are determined arbitrarily based on the wished performance, while the parameter p is defined in function of the other two parameters, so that the stability of the system is assured in accordance with *Lyapunov's equation* (3.7).

To determine the value of p , an auxiliary control law is defined so that the system is asymptotically stable

$$\mathbf{u}(k) = \mathbf{K} \mathbf{x}(k) \quad (7.6)$$

where K is the gain of the controller, defined through eigenvalue assignment, imposing that eigenvalues of the closed loop system are comprised between zero and one:

$$\mathbf{x}(k+1) = (\mathbf{A} + \mathbf{B} \mathbf{K}) \mathbf{x}(k)$$

Thus, K is a matrix of the form

$$\mathbf{K} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

where k is chosen arbitrarily chosen as

$$k = -\frac{1}{2\tau}$$

This choice of k fulfills the conditions of asymptotic stability and non-oscillating convergence, which are stated at (7.7) and (7.8), respectively:

$$k > -\frac{2}{\tau} \quad (7.7)$$

$$k > -\frac{1}{\tau} \quad (7.8)$$

Once k is computed, it is possible to determine the adequate value of p solving *Lyapunov's equation*:

$$p = \frac{q + k^2 r}{1 - (1 + \tau k)^2} \quad (7.9)$$

7.3 Constraints

As explained previously, one of the main characteristics of MPC control is the possibility of defining explicit constraints on the control signal to be obtained. Thus, the control signal computed by the controller will be the one minimizing the cost function while also fulfilling all the requirements.

In the case of study, several type of constraints need to be applied according to the different necessities of the control such as:

- Constraint the speed and acceleration to match (at least), the physical limitations of the actuators
- Stability of the control signal
- Restriction of the admissible region for the position, in order to satisfy the obstacle avoidance requirements
- Constraint the filtered acceleration (as explained in Chapter 6) to assure the comfort of the passenger

Therefore, this section will describe how all the necessary constraints, which can be lay out with respect to the state variables or control variables, are transformed so they can be written as linear expressions dependant on the control sequence and thus define a convex admissible region.

7.3.1 Velocity constraints

The linearization of the model does not allow to impose constraints on the original control variables v and ω . Because of this reason, the speed to be limited is this of the point \mathbf{P} of the feedback linearization.

In previous work [1], speed constraints were applied to the module of the speed $v_P = \sqrt{v_{Px}^2 + v_{Py}^2}$, but because of the non-linear relation, a linearization using the first term of *Taylor series* was necessary. In this project, however, this is not possible as another linearization is required for assuring ride comfort, and combining both leads to an unstable admissible area that diverges to infinity. For this reason, the velocity constraints defined are uncoupled, limiting each axis separately. Additionally, the maximum value of velocity v_{max} is divided by $\sqrt{2}$, as decoupling the constraints can result in both axes having the maximum speed, which would make the module higher than the maximum value

Therefore, the velocity constraints implemented are:

$$\begin{aligned} -\frac{v_{max}}{\sqrt{2}} &\leq v_{Px}(k+i) \leq \frac{v_{max}}{\sqrt{2}} \\ -\frac{v_{max}}{\sqrt{2}} &\leq v_{Py}(k+i) \leq \frac{v_{max}}{\sqrt{2}} \\ \forall i &\in \{0, 1, \dots, N-2\} \end{aligned} \tag{7.10}$$

To force the wheelchair to stop when convergence is reached, it is convenient to impose more strict constraints to the last iteration $k + N - 1$ of the velocity.

In this last iteration, the value of v_{max} is modified as the maximum variation of speed Δv_{max} . This variation is related to the maximum mean acceleration, which is defined arbitrarily in function of the physical limitations of the system. Given a maximum mean acceleration \bar{a}_{max} , Δv_{max} is computed as

$$\Delta v_{max} = \bar{a}_{max} \tau \quad (7.11)$$

In this way, it is possible to assure that the final speed transition before reaching the reference is not abrupt. The constraints applied are:

$$\begin{aligned} -\Delta v_{max} &\leq v_{Px}(k+N-1) \leq \Delta v_{max} \\ -\Delta v_{max} &\leq v_{Py}(k+N-1) \leq \Delta v_{max} \end{aligned} \quad (7.12)$$

The last step is to express the constraints in function of $\mathcal{U}(k)$ to be able to introduce them in the optimization solver. The constraints are expressed as:

$$\mathbf{A}_{v \ max} \mathcal{U}(k) \leq \mathbf{b}_{v \ max} \quad (7.13)$$

with

$$\mathbf{A}_{v \ max} = \begin{bmatrix} \mathbf{I}_{2N} \\ -\mathbf{I}_{2N} \end{bmatrix}_{(2N, 2N)} \quad \mathbf{b}_{v \ max} = \begin{bmatrix} v_{max}/\sqrt{2} \\ \vdots \\ v_{max}/\sqrt{2} \\ \Delta v_{max} \\ \Delta v_{max} \\ v_{max}/\sqrt{2} \\ \vdots \\ v_{max}/\sqrt{2} \\ \Delta v_{max} \\ \Delta v_{max} \end{bmatrix}_{(4N, 1)}$$

7.3.2 Position constraints

These constraints correspond to the convex admissible region explained in Chapter 5, which is defined considering all the detected obstacles in order to achieve the obstacle avoidance requirements. As explained before, the set of constraints computed is based on the belonging of the wheelchair into adequate semi-planes defined by each obstacle.

As an observation, due to the algorithm utilized the majority of them are redundant in the sense that some of them are less restrictive than others, so a limited set of the constraints used would be enough to define the admissible region.

For every obstacle detected, there exists a set of restrictions expressed as:

$$\begin{aligned} h_1 x_P(k+i) + h_2 y_P(k+i) &\leq l \\ \forall i \in \{0, 1, \dots, N\} \end{aligned} \quad (7.14)$$

or in compact form

$$\underbrace{\begin{bmatrix} h_1 & h_2 \end{bmatrix}}_{\mathbf{h}} \mathbf{x}(k+i) \leq l \quad (7.15)$$

$$\forall i \in \{0, 1, \dots, N\}$$

To ensure that the obstacle avoidance requirements are fulfilled through the whole prediction horizon it is necessary to impose a constraint for each instant in matrix form:

$$\underbrace{\begin{bmatrix} \mathbf{h}^{(i)} & & \\ & \ddots & \\ & & \mathbf{h}^{(i)} \end{bmatrix}}_{\mathbf{H}_{obs}^{(i)}} \underbrace{\begin{bmatrix} \mathbf{x}(k) \\ \vdots \\ \mathbf{x}(k+N) \end{bmatrix}}_{\mathbf{x}^{(k)}} \underbrace{\begin{bmatrix} l^{(i)} \\ \vdots \\ l^{(i)} \end{bmatrix}}_{\mathbf{L}_{obs}^{(i)}} \quad (7.16)$$

As mentioned in previous chapters, this constraint expression is not valid for the optimization solver, and has to be rewritten using *Lagrange's equation* (3.9) to express it in terms of $\mathbf{U}(k)$:

$$\mathbf{H}_{obs}^{(i)} \underbrace{(\mathcal{A} \mathbf{x}(k) + \mathcal{B} \mathbf{U}(k))}_{\mathbf{x}^{(k)}} \leq \mathbf{L}_{obs}^{(i)} \quad (7.17)$$

or

$$\mathbf{A}_{obs}^{(i)} \mathbf{U}(k) \leq \mathbf{b}_{obs}^{(i)} \quad (7.18)$$

with

$$\mathbf{A}_{obs}^{(i)} = \mathbf{H}_{obs}^{(i)} \mathcal{B} \quad \mathbf{b}_{obs}^{(i)} = \mathbf{L}_{obs}^{(i)} - \mathbf{H}_{obs}^{(i)} \mathcal{A} \mathbf{x}(k)$$

The whole set of obstacles can be expressed as the combination of the constraints for each obstacle as:

$$\mathbf{A}_{obs} = \begin{bmatrix} \mathbf{A}_{obs}^{(1)} \\ \vdots \\ \mathbf{A}_{obs}^{(n)} \end{bmatrix} \quad \mathbf{b}_{obs} = \begin{bmatrix} \mathbf{b}_{obs}^{(1)} \\ \vdots \\ \mathbf{b}_{obs}^{(n)} \end{bmatrix}$$

7.3.3 Ride comfort constraints

The aim of this set of constraints is to ensure the comfort of the passenger seated on the wheelchair. To achieve the desired level of comfort, it is necessary that the value of the *Overall Ride Value* [2] is limited according to ISO 2631-1. As explained in Chapter 6, the constraints to be imposed have the form:

$$ORV(k+i) = \frac{1}{\sqrt{2}} \sqrt{a_{w,Px}(k+i)^2 + a_{w,Py}(k+i)^2} \leq ORV_{max} \quad (7.19)$$

$$\forall i \in \{0, 1, \dots, N-1\}$$

The filtered acceleration $a_{w,i}$ for each axis x and y is related to past values of linear acceleration and the filtered acceleration itself. Given the discrete transfer function relating the filtered and linear accelerations

$$\frac{A_w(z)}{A(z)} = H(z) = \frac{\sum_{j=0}^m c_j z^j}{\sum_{j=0}^n b_j z^j} \quad m \leq n \quad (7.20)$$

The expression of filtered acceleration is

$$a_w(k) = \frac{-\sum_{j=1}^{n-1} b_j a_w(k-n+j) + \sum_{j=0}^m c_j a(k-m+j)}{b_n} \quad (7.21)$$

with

$$a(k) = \frac{v_P(k) - v_P(k-1)}{\tau}$$

Therefore, the objective is to express the constraints defined in (7.19) with respect to v_{Px} and v_{Py} in order to include them into the optimization problem. As can be seen, the constraints to be applied present a non-linear relation of the variables, because although if the definition of the weighted acceleration in one axes consists of linear combinations of previous v_P , the *ORV* definition presents non-linearity.

In order to deal with this problem, the expression of *ORV* is linearized near the last iteration acceleration utilizing the first terms of *Taylor series*

$$0 \leq \sqrt{\bar{a}_{wx}^2 + \bar{a}_{wy}^2} + \frac{1}{2 \sqrt{\bar{a}_{wx}^2 + \bar{a}_{wy}^2}} (2 \bar{a}_{wx} (a_{wx} - \bar{a}_{wx}) + 2 \bar{a}_{wy} (a_{wy} - \bar{a}_{wy})) \leq \sqrt{2} ORV_{max} \quad (7.22)$$

which can be rewritten as

$$0 \leq \frac{\bar{a}_{wx}}{\bar{a}_w} a_{wx}(k+i) + \frac{\bar{a}_{wy}}{\bar{a}_w} a_{wy}(k+i) \leq \sqrt{2} ORV_{max} \quad (7.23)$$

$$\forall i \in \{0, 1, \dots, N-1\}$$

with $\bar{a}_w = \sqrt{\bar{a}_{wx}^2 + \bar{a}_{wy}^2}$ the acceleration on the previous iteration.

However, this expression is not applicable for values of weighted acceleration near zero, as the term \bar{a}_w is located in the denominator. For this reason, in the case of low acceleration, the filtered acceleration constraints are uncoupled and applied to each axis separately:

$$\begin{aligned}
-ORV_{max} &\leq a_{wx}(k+i) \leq ORV_{max} \\
-ORV_{max} &\leq a_{wy}(k+i) \leq ORV_{max} \\
\forall i &\in \{0, 1, \dots, N-1\}
\end{aligned} \tag{7.24}$$

The notation for this kind of constraints with respect to v_{Px} and v_{Py} in matrix form is:

$$\mathbf{N}(\mathbf{M}\mathbf{U}(k) - \bar{\mathbf{a}}_w(k)) \leq \mathbf{RV}_{max} \tag{7.25}$$

with

$$\begin{aligned}
\mathbf{N} &= \begin{bmatrix} \mathbf{I}_{2N} \\ -\mathbf{I}_{2N} \end{bmatrix}_{(4N, 2N)} & \mathbf{N} &= \begin{bmatrix} \mathbf{I}_N \\ -\mathbf{I}_N \end{bmatrix}_{(2N, N)} \\
\mathbf{RV}_{max} &= \begin{bmatrix} ORV_{max} \\ \vdots \\ ORV_{max} \end{bmatrix}_{(4N, 1)} & \mathbf{RV}_{max} &= \begin{bmatrix} \sqrt{2} ORV_{max} \\ \vdots \\ \sqrt{2} ORV_{max} \end{bmatrix}_{(2N, 1)} \\
&\text{For low accelerations} & & \text{For high accelerations}
\end{aligned}$$

Because the relation between acceleration and weighted acceleration depends on previous instants, it is necessary to generate the matrix iteratively. The process for obtaining the matrices \mathbf{M} and $\bar{\mathbf{a}}_w(k)$ is explained below, while the implementation can be seen in Algorithms 2 and 3.

Matrix \mathbf{M}

This matrix contains the relation between the control variables v_{Px} , v_{Py} and the filtered acceleration. As stated before, due to the nature of the filtered acceleration, every step in the prediction horizon requires a combination of all previous accelerations, which results in a relation with a difficult analytic expression. This matrix depends only on the coefficients of the filter and thus can be computed before starting the control iterations.

The first step is to set a matrix \mathbf{M}_1 which considers only the part related to the linear acceleration at each instant, i.e. the terms containing the coefficients $\{c_m, \dots, c_0\}$. This matrix can be defined as:

$$\mathbf{M}_1 = \frac{1}{\tau b_n} \begin{bmatrix} c_m \mathbf{I}_2 & & & & & \\ (c_{m-1} - c_m) \mathbf{I}_2 & c_m \mathbf{I}_2 & & & & \\ \vdots & (c_{m-1} - c_m) \mathbf{I}_2 & & & & \\ (c_0 - c_1) \mathbf{I}_2 & \vdots & \ddots & & & \\ -c_0 \mathbf{I}_2 & (c_0 - c_1) \mathbf{I}_2 & \ddots & \ddots & & \\ & -c_0 \mathbf{I}_2 & \ddots & \ddots & & \\ & & \vdots & \vdots & & \\ & & & c_m \mathbf{I}_2 & & \\ & & & (c_{m-1} - c_m) \mathbf{I}_2 & c_m \mathbf{I}_2 & \end{bmatrix}_{(2N, 2N)}$$

The next step consists of, given a pair rows, subtract the $n - 1$ previous couple of rows recursively (if all the rows are available), multiplied by the adequate coefficient of filtered acceleration. It is important to note that it is necessary to divide every coefficient $\{b_{n-1}, \dots, b_0\}$ by b_n previously, as b_n is common to every term. A simple example is shown below considering the first rows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1(1:2,:) \\ \mathbf{M}_1(3:4,:) - b_{n-1}/b_n \underbrace{(\mathbf{M}_1(1:2,:))}_{\mathbf{M}(1:2,:)} \\ \mathbf{M}_1(5:6,:) - b_{n-1}/b_n \underbrace{(\mathbf{M}_1(3:4,:) - b_{n-1}/b_n \underbrace{\mathbf{M}_1(1:2,:)}_{\mathbf{M}(1:2,:)})}_{\mathbf{M}(3:4,:)} - b_{n-2}/b_n \underbrace{(\mathbf{M}_1(1:2,:))}_{\mathbf{M}(1:2,:)} \\ \vdots \end{bmatrix}_{(2N, 2N)}$$

where $\mathbf{M}(i:j,:)$ refers to all the elements of rows i to j of the matrix \mathbf{M}

The algorithm used, considering that the vectors containing the coefficients are sorted backwards ($a_{coefs} = [c_m, \dots, c_0]$; $aw_{coefs} = [b_n, \dots, b_0]$)

Algorithm 2: $m_mat(a_{coefs}, aw_{coefs}, N, T)$:

```

 $l_a = \text{length}(a_{coefs});$ 
 $a_{coefs} = a_{coefs}/aw_{coefs}[0];$ 
 $aw_{short} = aw_{coefs}[1 : \text{end}]/aw_{coefs}[0];$ 
 $l_{aw} = \text{length}(aw_{short});$ 

 $\mathbf{M}_1 = a_{coefs}[0] * \text{identity}(2 * N, 2 * N);$ 
for  $i = 1$  to  $l_a - 1$  do
     $\mathbf{M}_{it} = (a_{coefs}[i] - a_{coefs}[i - 1]) * \text{identity}(2 * (N - i), 2 * (N - i));$ 
     $\mathbf{M}_1 = \mathbf{M}_1 + [\text{zeros}(2 * i, 2 * (N - i)); \mathbf{M}_{it}, \text{zeros}(2 * N, 2 * i)];$ 
end

 $\mathbf{M}_{last} = -a_{coefs}[l_a] * [\text{zeros}(2 * l_a, 2 * (N - l_a)); \text{identity}(2 * (N - l_a)), \text{zeros}(2 * N, 2 * l_a)];$ 
 $\mathbf{M}_1 = (1/(aw_{coefs}[0] * T)) * (\mathbf{M}_1 + \mathbf{M}_{last});$ 
 $\mathbf{M} = \mathbf{M}_1[0 : 1, :];$ 

for  $i = 2$  to  $2 * (N - 1)$  by  $2$  do
    if  $i < 2 * l_a$  then
        // All required previous rows cannot be computed
        for  $k = 0$  to  $i/2$  do
             $\mathbf{M} = \mathbf{M}_1[i : i + 1, :] - aw_{short}[k] * \mathbf{M}[i - 2 * k - 1 : i - 2 * k, :];$ 
        end
    else
        // All required previous rows can be computed
        for  $k = 0$  to  $l_{aw} - 1$  do
             $\mathbf{M} = \mathbf{M}_1[i : i + 1, :] - aw_{short}[k] * \mathbf{M}[i - 2 * k - 1 : i - 2 * k, :];$ 
        end
    end
end

```

Vector $\bar{\mathbf{a}}_w(k)$

This vector contains the terms of the filtered acceleration constraints that not depend on the control variables, but on stored values from previous instants. As every iteration a new value of the filtered acceleration is computed, this vector changes at every iteration of the control loop.

The strategy adopted in this case is similar to the one utilized for the M matrix explained above. There are two different situations to consider when filling the vector $\bar{\mathbf{a}}_w(k)$:

- Direct application of the expression with known values
- Product with coefficients of previous instants as a consequence of the recursive nature of the constraints

Thus, the first step is to define a vector $\bar{\mathbf{a}}_{w1}$ considering less of the known values each instant in the form:

$$\bar{\mathbf{a}}_{w1}(k) = \begin{bmatrix} -\sum_{j=1}^{n-1} b_j a_{wx}(k-n+j) + \sum_{j=0}^{m-1} c_j a_x(k-m+j) - \frac{c_m}{\tau} v_{Px}(k-1) \\ -\sum_{j=1}^{n-1} b_j a_{wy}(k-n+j) + \sum_{j=0}^{m-1} c_j a_y(k-m+j) - \frac{c_m}{\tau} v_{Py}(k-1) \\ -\sum_{j=1}^{n-2} b_j a_{wx}(k-n+j-1) + \sum_{j=1}^{m-2} c_j a_x(k-m+j-1) - \frac{c_{m-1}}{\tau} v_{Px}(k-1) \\ -\sum_{j=1}^{n-2} b_j a_{wy}(k-n+j-1) + \sum_{j=1}^{m-2} c_j a_y(k-m+j-1) - \frac{c_{m-1}}{\tau} v_{Py}(k-1) \\ \vdots \\ -c_0 v_{Px}(k-1) \\ -c_0 v_{Py}(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2N,1)$$

in the case of $n = m$.

Once this vector is computed, as in the case of the matrix M , each couple of rows is multiplied by the adequate coefficients of weighted acceleration recursively. A simple example considering the first values is shown below:

$$\bar{\mathbf{a}}_w = \begin{bmatrix} \bar{\mathbf{a}}_{w1}[0:1] \\ \bar{\mathbf{a}}_{w1}[2:3] - b_{n-1}/b_n \underbrace{(\bar{\mathbf{a}}_{w1}[0:1])}_{\bar{\mathbf{a}}_w[0:1]} \\ \bar{\mathbf{a}}_{w1}[4:5] - b_{n-1}/b_n \underbrace{(\bar{\mathbf{a}}_{w1}[2:3] - b_{n-1}/b_n (\bar{\mathbf{a}}_{w1}[0:1]))}_{\bar{\mathbf{a}}_w[0:1]} - b_{n-2}/b_n \underbrace{(\bar{\mathbf{a}}_{w1}[0:1])}_{\bar{\mathbf{a}}_w[0:1]} \\ \vdots \end{bmatrix} \quad (2N,1)$$

Below is explained the algorithm implemented for computing $\bar{\mathbf{a}}_w(k)$. Again, the coefficient arrays are sorted backwards ($a_{coefs} = [c_m, \dots, c_0]$; $aw_{coefs} = [b_n, \dots, b_0]$). The vectors a_{prev} , aw_{prev} and v_{prev} contain previous values of the acceleration, the filtered acceleration and the velocity of the point P respectively. All vectors are sorted so that the first value of the vector is the most recent measure.

Algorithm 3: $aw_bar(a_{coefs}, aw_{coefs}, N, T, a_{prev}, aw_{prev}, v_{prev})$:

```

 $l_a = \text{length}(a_{coefs});$ 
 $a_{coefs} = a_{coefs}/aw_{coefs}[0];$ 
 $aw_{short} = aw_{coefs}[1 : \text{end}]/aw_{coefs}[0];$ 
 $l_{aw} = \text{length}(aw_{short});$ 
 $a_{bar} = \text{zeros}(2 * N, 1);$ 

// Known values of linear acceleration
 $k = 0;$ 
for  $i = 0$  to  $l_a - 2$  do
     $a_{bar}[2 * i] = a_{bar}[2 * i] - (a_{coefs}[i]/T) * v_{prev}.x[1];$ 
     $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] - (a_{coefs}[i]/T) * v_{prev}.y[1];$ 
    if  $i < l_{aw} - 2$  then
        for  $j = i + 1$  to  $l_{aw} - 2$  do
             $a_{bar}[2 * i] = a_{bar}[2 * i] + a_{coefs}[i] * a_{prev}.x[k_a] - aw_{short}[i] * aw_{prev}.x[k_{aw}];$ 
             $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] + a_{coefs}[i] * a_{prev}.y[k_a] - aw_{short}[i] * aw_{prev}.y[k_{aw}];$ 
             $k = k + 1;$ 
        end
    end
     $k = 0;$ 
end

// Known values of filtered acceleration
 $k = 1;$ 
for  $i = 0$  to  $l_{aw} - 2$  do
     $a_{bar}[2 * i] = a_{bar}[2 * i] - aw_{short}[i] * aw_{prev}.x[1];$ 
     $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] - aw_{short}[i] * aw_{prev}.y[1];$ 
    if  $i < l_{aw} - 2$  then
        for  $j = i + 1$  to  $l_{aw} - 2$  do
             $a_{bar}[2 * i] = a_{bar}[2 * i] - aw_{short}[i] * aw_{prev}.x[k_{aw}];$ 
             $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] - aw_{short}[i] * aw_{prev}.y[k_{aw}];$ 
             $k = k + 1;$ 
        end
    end
     $k = 1;$ 
end

// Completion of the instants within the prediction horizon
for  $i = 1$  to  $l_{aw} - 1$  do
    for  $j = 0$  to  $i - 1$  do
         $a_{bar}[2 * i] = a_{bar}[2 * i] - aw_{short}[j] * a_{bar}[2 * (i - j - 1)];$ 
         $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] - aw_{short}[j] * a_{bar}[2 * (i - j - 1)];$ 
    end
end

for  $i = l_{aw}$  to  $N - 1$  do
    for  $j = 0$  to  $l_{aw} - 1$  do
         $a_{bar}[2 * i] = a_{bar}[2 * i] - aw_{short}[j] * a_{bar}[2 * (i - j - 1)];$ 
         $a_{bar}[2 * i + 1] = a_{bar}[2 * i + 1] - aw_{short}[j] * a_{bar}[2 * (i - j - 1)];$ 
    end
end
end

```

Coupling of the constraint matrices

After the process to determine \mathbf{M} and $\bar{\mathbf{a}}_w(k)$ described by Algorithms 2 and 3, the result is the matrix form of the decoupled constraints defined in (7.24). In order to obtain the linear expression of the ORV , defined in (7.23) and showed below for more clarity

$$0 \leq \frac{\bar{a}_{wx}}{\bar{a}_w} a_{wx}(k+i) + \frac{\bar{a}_{wy}}{\bar{a}_w} a_{wy}(k+i) \leq \sqrt{2} ORV_{max} \quad (7.26)$$

it is necessary to:

- Calculate the coefficients $\frac{\bar{a}_{wx}}{\bar{a}_w}$ and $\frac{\bar{a}_{wy}}{\bar{a}_w}$ based on the filtered acceleration of the previous iteration
- Combine even and odd rows of \mathbf{M} and $\bar{\mathbf{a}}_w(k)$, multiplying them by the adequate coefficient

This can be done easily as shown in Algorithm 4.

Algorithm 4: *comfort_coupling*($\mathbf{M}, \mathbf{a}_{bar}, N, aw_{kprev}$)

```

lin_coefs = aw_kprev/norm(aw_kprev);
M_high = zeros(N, N); a_bar_high = zeros(N, 1) for i = 0 to N - 1 do
    M_high[i, :] = lin_coefs[0] * M[2 * i, :] + lin_coefs[1] * M[2 * i + 1, :];
    a_bar_high[i] = lin_coefs[0] * a_bar[2 * i] + lin_coefs[1] * a_bar[2 * i + 1];
end

```

Once the matrices \mathbf{M} and $\bar{\mathbf{a}}(k)$ have been computed, the constraints are defined as stated in (7.25), or in compact form:

$$\mathbf{A}_{com} \mathbf{U}(k) \leq \mathbf{b}_{com} \quad (7.27)$$

with:

$$\mathbf{A}_{com} = \mathbf{N} \mathbf{M} \quad \mathbf{b}_{com} = \mathbf{RV}_{max} - \mathbf{N} \bar{\mathbf{a}}_w(k)$$

For low accelerations, these matrices correspond to

$$\mathbf{N} = \begin{bmatrix} \mathbf{I}_{2N} \\ -\mathbf{I}_{2N} \end{bmatrix}_{(4N, 2N)} \quad \mathbf{RV}_{max} = \begin{bmatrix} ORV_{max} \\ \vdots \\ ORV_{max} \end{bmatrix}_{(4N, 1)}$$

$$\mathbf{M}_{(2N, 2N)} \text{ as described in Alg. 2} \quad \bar{\mathbf{a}}_w(k)_{(2N, 1)} \text{ as described in Alg. 3}$$

whereas for high accelerations, the matrix to be utilized are

$$\mathbf{N} = \begin{bmatrix} \mathbf{I}_N \\ -\mathbf{I}_N \end{bmatrix}_{(2N, N)} \quad \mathbf{RV}_{max} = \begin{bmatrix} \sqrt{2} ORV_{max} \\ \vdots \\ \sqrt{2} ORV_{max} \end{bmatrix}_{(2N, 1)}$$

$$\mathbf{M}_{(N, 2N)} \text{ as described in Alg. 4} \quad \bar{\mathbf{a}}_w(k)_{(N, 1)} \text{ as described in Alg. 4}$$

7.4 Vortex Field

The position constraints implemented do not guarantee an effective obstacle avoidance performance throughout the control process. For example, in the case that the trajectory minimizing the cost function does not consider avoiding the obstacle, preventing the solver to converge.

In order to solve this issue a common technique is to consider, besides the attractive field towards the reference, a repulsive force field to affect the trajectory generated and allow the vehicle to avoid the obstacle.

Adding this vortex field to the cost function is possible, however this results in a cost function that not satisfies the generalized quadratic problem expression, thus increasing the computational time and not allowing a real-time implementation.

The solution applied in this project is to provide an adequate reference to the controller when an obstacle is located within a distance lower than a threshold, so that the vehicle changes its direction to avoid the obstacle while allowing a correct evolution of the admissible region imposed in the position constraints.

Therefore, to fulfill the obstacle avoidance requirements, an artificial repulsive field has been utilized, which is activated when the vehicle is close to an obstacle and changes the reference position in order to avoid the object. This field is inversely proportional to the distance of the wheelchair respect the object, in order to exert a bigger influence as the vehicle approaches the obstacle.

7.4.1 Application of the vortex field in the MPC problem

The classical formulation of a vortex field consists of defining a vortex vector

$$\nabla \mathbf{F}_{vor}(k) = \begin{bmatrix} \nabla x_{vor}(k) \\ \nabla y_{vor}(k) \end{bmatrix} \quad (7.28)$$

in order to produce a rotation around the obstacle to be avoided, with respect to the current position of the vehicle.

This vector is applied to the current position to calculate the new vortex field reference which the vehicle has to follow:

$$\mathbf{x}_{vor}(k) = \mathbf{x}(k) + \nabla \mathbf{F}_{vor}(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} + \begin{bmatrix} \nabla x_{vor}(k) \\ \nabla y_{vor}(k) \end{bmatrix} \quad (7.29)$$

To ensure that the obstacle avoidance are fulfilled satisfactorily, the vortex vortex has to meet the following requirements:

- $\|\nabla \mathbf{F}_{vor}(k)\| \rightarrow \infty$ when approaching the obstacle
- The influence of the field has to be limited as not to disrupt the vehicle behaviour when it is not sufficiently close to the obstacle.
- Its evolution has to be continuous respect the position

The field vortex is included into the optimization problem through the cost function, modifying the parameters adequately at each iteration if the activation of the vortex is required. The parameters modified for the implementation are \mathbf{x}_{ref} , \mathbf{Q} , \mathbf{P} , which are changed in the new cost function

$$J(k) = \sum_{i=0}^N \left(\|\mathbf{x}(k+i) - \mathbf{x}'_{ref}\|_{\mathbf{Q}'}^2 + \|\mathbf{u}(k+i)\|_{\mathbf{R}}^2 \right) + \|\mathbf{x}(k+N) - \mathbf{x}'_{ref}\|_{\mathbf{P}'}^2 \quad (7.30)$$

with \mathbf{x}'_{ref} , \mathbf{Q}' and \mathbf{P}' changing based on the necessity of activating the vortex field. Precisely:

$$\mathbf{x}'_{ref} = \begin{cases} \mathbf{x}_{vor} & \text{if the vortex field is active} \\ \mathbf{x}_{ref} & \text{otherwise} \end{cases}$$

$$\mathbf{Q}' = \begin{cases} \mathbf{Q}_{vor} = \begin{bmatrix} q_{vor} & 0 \\ 0 & q_{vor} \end{bmatrix} & \text{if the vortex field is active} \\ \mathbf{Q} & \text{otherwise} \end{cases}$$

with $q_{vor} \gg q$ to highly penalize the error the state and reach the vortex reference as fast as possible. The terminal state error \mathbf{P}' is

$$\mathbf{P}' = \begin{cases} \mathbf{P}_{vor} = \begin{bmatrix} p_{vor} & 0 \\ 0 & p_{vor} \end{bmatrix} & \text{if the vortex field is active} \\ \mathbf{P} & \text{otherwise} \end{cases}$$

where p_{vor} is calculated following as explained in previous chapters utilizing *Riccati's equation* with the new value of q_{vor} .

Thus, the main idea is to change the reference and weight of the error of state variables when any obstacle is detected at a certain distance of the vehicle, to then go back to the original optimization problem based on the desired reference when the obstacle is surpassed.

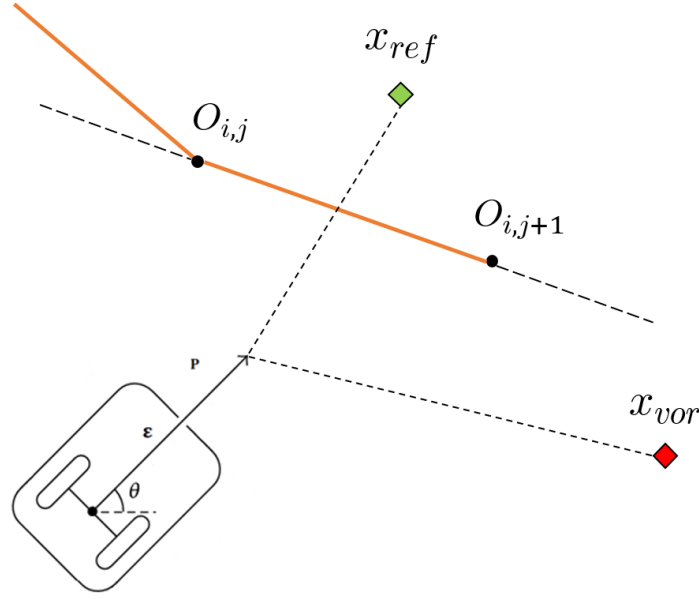


Figure 7.1: A different reference is set to avoid the obstacle

The application of this type of field can involve undesired variations on the velocity of the vehicle. For this reason it is important that the vortex vector module is high enough in order to achieve smoother transitions on the trajectory. The minimum module of $\nabla \mathbf{F}_{vor}(k)$ is defined as the extreme case where the vehicle is traveling at the maximum speed v_{max} . i.e. $v_{Px} = v_{Py} = v_{max}/\sqrt{2}$, which corresponds to the minimum distance needed to stop the vehicle at constant deceleration.

$$d_{stop} = \frac{\tau v_{max}^2}{\Delta v_{max}}$$

Therefore, the condition required for the module of $\nabla \mathbf{F}_{vor}(k)$ is

$$\|\nabla \mathbf{F}_{vor}(k)\| > d_{stop} \quad (7.31)$$

7.5 Activation and calculation of the vortex field vector

As stated previously, the objective is to activate the vortex field only in the case that an obstacle is detected close to the vehicle. The vortex vector varies depending on the position of the wheelchair with respect the obstacle, and thus has to be calculated every iteration where the vortex field is activated.

Obstacles are detected based on the obstacle segmentation explained in Chapter 5, and the lines that define them are the same utilized to construct the obstacle avoidance constraints.

The line between the vehicle and the reference can be defined as

$$h_{1ref} x + h_{2ref} y = l_{ref}$$

with

$$\begin{cases} h_{1ref} = y_{ref} - y_P \\ h_{2ref} = x_P - x_{ref} \\ l_{ref} = h_{1ref} x_P + h_{2ref} y_P \end{cases}$$

This line is compared iteratively with the lines defined by every couple of points in the list of obstacles. For every two consecutive points of the obstacle i , $\mathbf{O}_{i,j}$ and $\mathbf{O}_{i,j+1}$, the line between them can be expressed as

$$h_{1obs} x + h_{2obs} y = l_{obs}$$

with

$$\begin{cases} h_{1obs} = y_{i,j+1} - y_{i,j} \\ h_{2obs} = x_{i,j} - x_{i,j+1} \\ l_{obs} = h_{1obs} x_{i,j} + h_{2obs} y_{i,j} \end{cases}$$

From this point, it is possible to calculate the intersection between both lines as

$$\mathbf{P}_{int} = \begin{cases} x_{int} = \frac{h_{2ref} \cdot l_{obs} - h_{2obs} \cdot l_{ref}}{h_{2ref} \cdot h_{1obs} - h_{2obs} \cdot h_{1ref}} \\ y_{int} = \frac{h_{1ref} \cdot l_{obs} - h_{1obs} \cdot l_{ref}}{h_{1ref} \cdot h_{2obs} - h_{1obs} \cdot h_{2ref}} \end{cases} \quad (7.32)$$

Once the intersection point between the lines is determined, it is necessary to check if the trajectory to the reference is blocked by the obstacle or the vehicle can move freely. To achieve this, the following conditions are examined

$$\mathbf{P}_{int} \quad s.t. \quad \begin{cases} \min(x_P, x_{ref}) \leq x_{int} \leq \max(x_P, x_{ref}) \\ \min(y_P, y_{ref}) \leq y_{int} \leq \max(y_P, y_{ref}) \\ \min(x_{i,j}, x_{i,j+1}) \leq x_{int} \leq \max(x_{i,j}, x_{i,j+1}) \\ \min(y_{i,j}, y_{i,j+1}) \leq y_{int} \leq \max(y_{i,j}, y_{i,j+1}) \end{cases} \quad (7.33)$$

In the case that the condition is not fulfilled for none of the two segments $\overrightarrow{\mathbf{P} \mathbf{x}_{ref}}$ and $\overrightarrow{\mathbf{O}_{i,j} \mathbf{O}_{i,j+1}}$, then it is considered that the path between the vehicle and the reference is not blocked.

However, if the condition is fulfilled for one of the segments (or both, as seen in Figure 7.2), then the path is blocked and the activation of the vortex field is required. This can be assured as the only intersection because the data is assumed to be measured radially, and thus if the intersection point belongs to the area described there is only one possible obstacle in that direction.

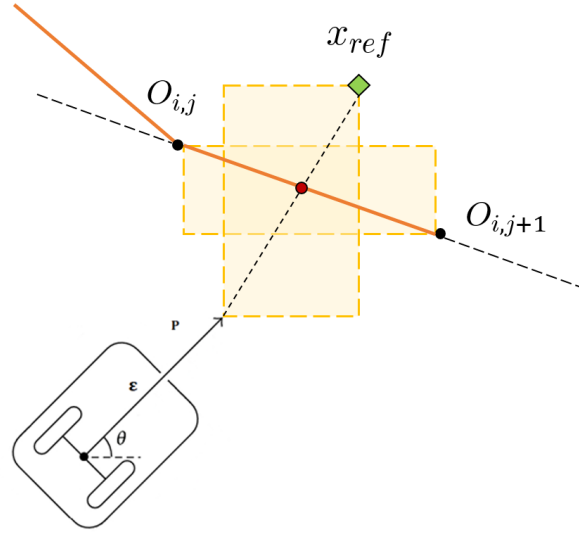


Figure 7.2: Intersection inside at least one area defined by the segments

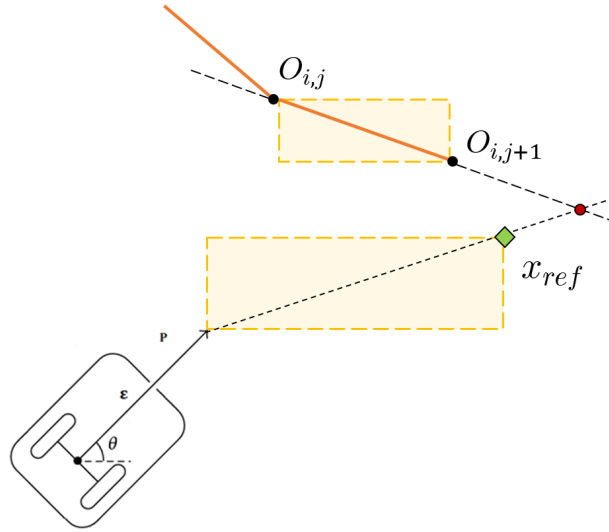


Figure 7.3: Intersection outside both areas defined by the segments

The direction of $\nabla \mathbf{F}_{vor}(k)$ is chosen based on the position constraint associated with the obstacle which contains the segment that blocks the path of the vehicle, whose choice is described in Section 5.4.

Given the constraint line

$$h_{1st} x + h_{2,st} y = l_{st}$$

the condition for the vortex field activation is that the line is located at a distance lower than an arbitrarily set threshold:

$$d(\mathbf{P}, \mathbf{r}_{st}) \leq d_{vor}$$

The direction of the evasion, respect the global axes, is chosen as the closest one to the current orientation

$$\theta_{vor}(k) = \begin{cases} \arctan\left(-\frac{h_{1st}}{h_{2st}}\right) & \text{if } \left\| \theta(k) - \arctan\left(-\frac{h_{1st}}{h_{2st}}\right) \right\| \leq \frac{\pi}{2} \\ \arctan\left(-\frac{h_{1st}}{h_{2st}}\right) + \pi & \text{otherwise} \end{cases} \quad (7.34)$$

Regarding the module of the vortex vector, its value is computed inversely proportional to the distance between the vehicle and the obstacle, including also some parameters which can be set arbitrarily depending on the control requirements:

$$\|\nabla \mathbf{F}_{vor}(k)\| = \nabla F_{min} \frac{d_{vor}}{d(k)} \quad (7.35)$$

This module of the vortex vector fulfills the requirements stated at the beginning of the chapter, as d_{vor} limits its range of action and because of the inverse proportionality with respect to $d(k)$ it tends to infinite when approaching the obstacle. Additionally, inside the influence region the position evolution is continuous.

At last, the vortex field reference $x_{vor}(k)$ is expressed as

$$\mathbf{x}_{vor}(k) = \begin{bmatrix} x_P(k) \\ y_P(k) \end{bmatrix} + \|\nabla \mathbf{F}_{vor}\| \begin{bmatrix} \cos \theta_{vor}(k) \\ \sin \theta_{vor}(k) \end{bmatrix} \quad (7.36)$$

Chapter 8

Simulations

This chapter describes the results of the simulations based on the now completely defined optimization problem required for applying the MPC technique.

The objective of the simulations is to implement the controller software that, given the position of the feedback linearization point \mathbf{P} , determines which is the optimal reference velocity to apply to the system in the next iteration.

The system used for simulation can be divided in two parts:

- **MPC controller:** At each instant, computes the necessary values for the constraint matrices and solves the optimization problem explained at 7. The controller is implemented using *MATLAB*.
- **Complete model:** The control signal determined at the previous step is introduced into the complete system, which includes the feedback linearization and the unicycle model, and the response of the system is sent back to the controller for further calculations. The complete model has been implemented using *Simulink*

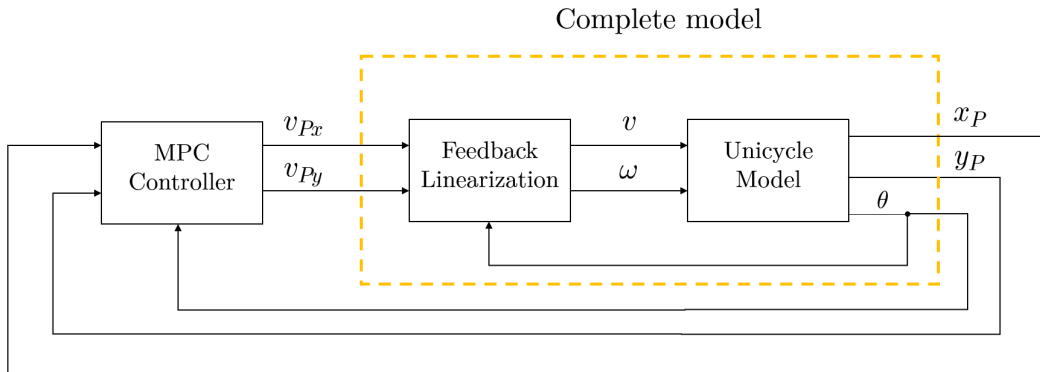


Figure 8.1: Control scheme utilized in the simulations

8.1 Common parameters in the simulations

During the course of the whole set of simulations performed, some of the parameters which were presented in previous chapter do not vary. For this reason, a list with each one of these parameters and their value is presented in this section.

Cost function	Control	Limitations	Vortex activation
$q = 1$	$\tau = 0.2 \text{ s}$	$v_{max} = 0.55 \text{ m/s}$	$\nabla F_{min} = 2.5 \text{ m}$
$r = 5$	$N = 15$	$a_{max} = 0.2 \text{ m/s}^2$	$d_{vor} = 1.7 \text{ m}$
$q_{vor} = 10 \cdot q$	$\varepsilon = 0.3 \text{ m}$	$ORV_{max} = 0.315 \text{ m/s}^2$	

Table 8.1: Parameters of transfer functions of the additional frequency weightings

8.2 Convergence towards a reference

The first simulation consists on verifying the most simple control requirement, which is the convergence from the initial point towards a reference point P_{ref} .

In the case tested, the vehicle parts from $P_0 = (0, 0)$ and the reference is set to $P_{ref} = (6, 3)$. For this simulation, only the velocity and ride comfort constraints are applied, as there is no obstacle present.

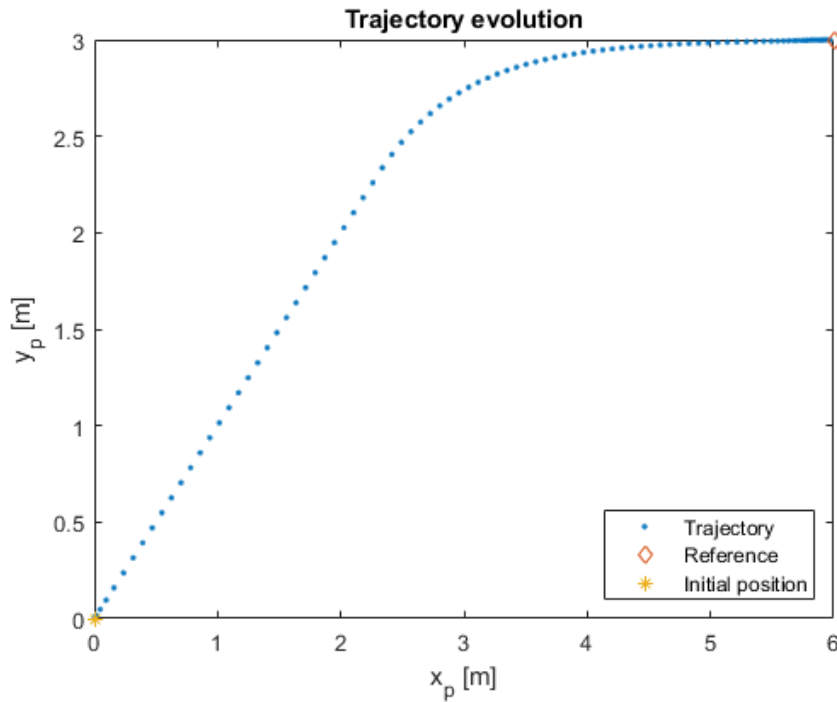


Figure 8.2: Convergence towards a reference

As can be seen in Figures 8.3 and 8.4, the limitations imposed are fulfilled adequately.

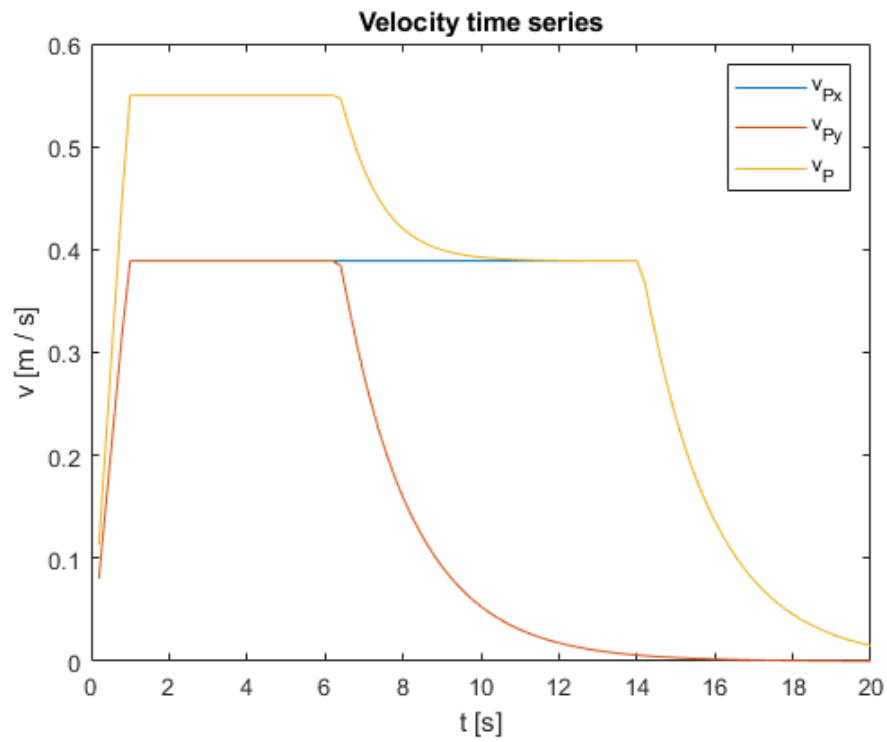


Figure 8.3: Evolution of the velocity

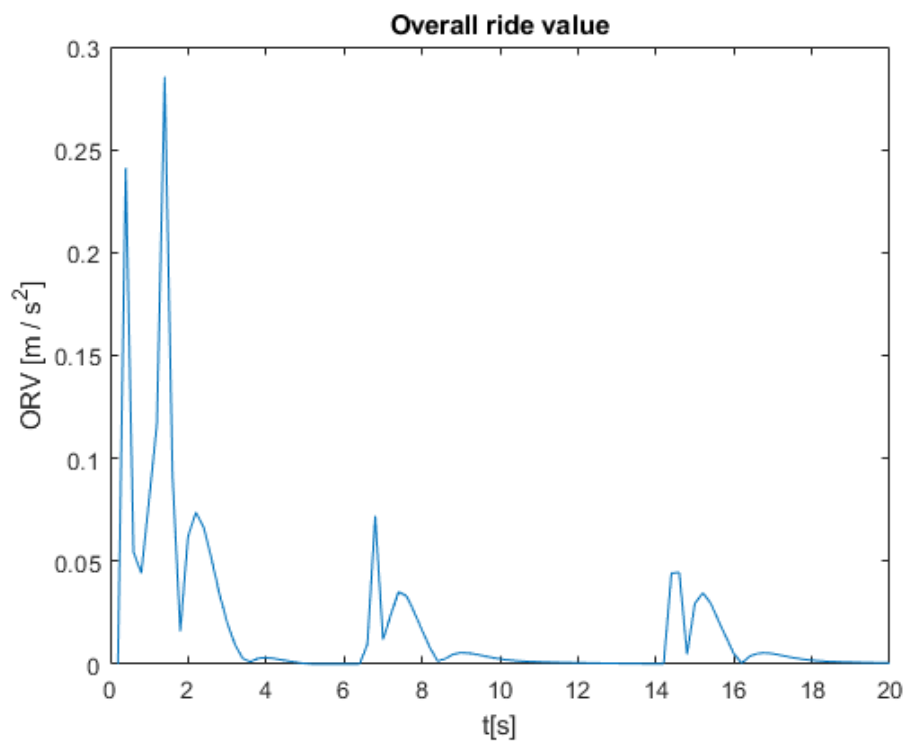


Figure 8.4: Overall ride value

Regarding velocity, it can be observed that the velocity profile results as desired, traveling at maximum speed with smooth start and ending transitions. As stated before, the velocity has been constrained separately for each axis for a value $v_{max}/\sqrt{2}$, to achieve what can be seen at the first instants: when each axis presents its maximum speed, the module is $v_{max} = 0.55 \text{ m/s}$

As for the ride comfort, the values of ORV are below the maximum $ORV_{max} = 0.315 \text{ m/s}^2$ at each instant, so the ride comfort is guaranteed.

8.3 Verification of the position constraints

To verify the validity of the constraints applied to the state variables, it is possible to evaluate the behaviour of the wheelchair when the reference is set outside of its admissible region.

The test consists on creating an octagonal admissible region centered in the initial position, and apply a variable reference that describes a circle also centered in the initial position.

The octagonal admissible region can be defined as

$$\begin{aligned} x_P \leq d, \quad -x_P \leq d, \quad y_P \leq d, \quad -y_P \leq d, \\ x_P + y_P \leq \sqrt{2}d, \quad x_P - y_P \leq \sqrt{2}d, \quad -x_P + y_P \leq \sqrt{2}d, \quad -x_P - y_P \leq \sqrt{2}d. \end{aligned}$$

and the circular reference is

$$\mathbf{P}_{ref}(k) = \begin{cases} x_{ref}(k) = r_{ref} \cos(2\pi \frac{k}{k_{2\pi}}) \\ y_{ref}(k) = r_{ref} \sin(2\pi \frac{k}{k_{2\pi}}) \end{cases}$$

As at this point the objective is to determine if the vehicle is able to stay within an admissible region, the vortex field is not applied in this simulation. The parameters used for this simulation are:

$$P_0 = (0, 0), \quad d = 3 \text{ m}, \quad r_{ref} = 4 \text{ m}, \quad k_{2\pi} = 200$$

The simulation time has been $t = 40 \text{ s}$, which is enough to traverse the majority of the region considering the constraints on speed and acceleration.

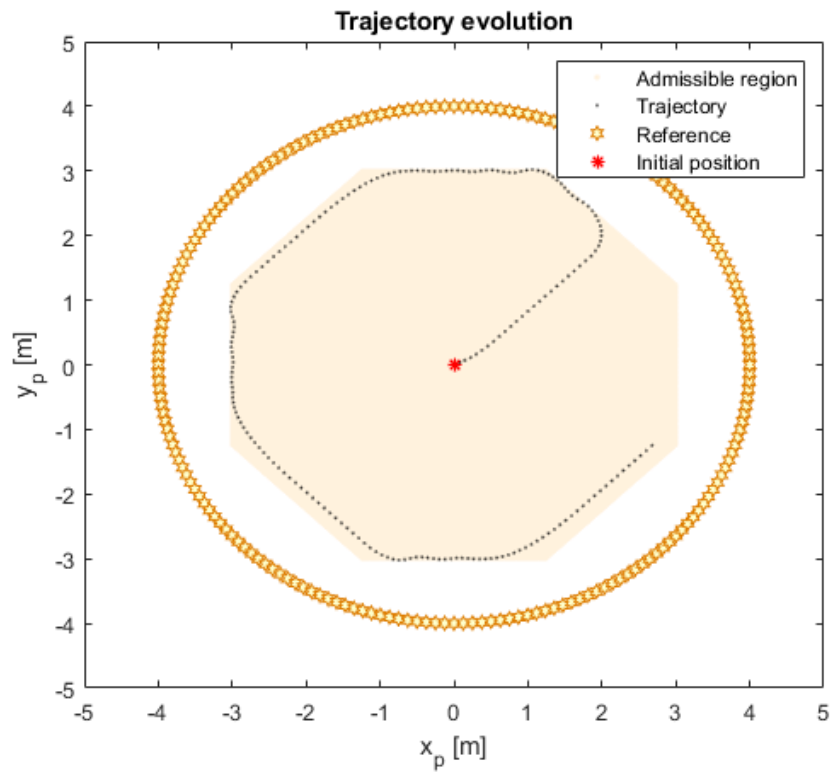


Figure 8.5: Trajectory within the octagonal area

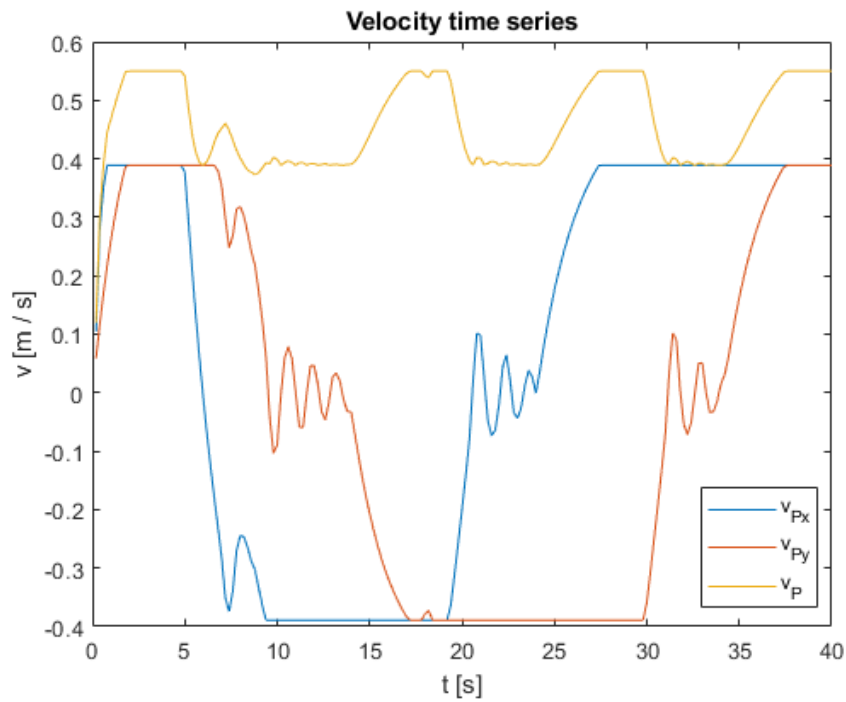


Figure 8.6: Evolution of the velocity

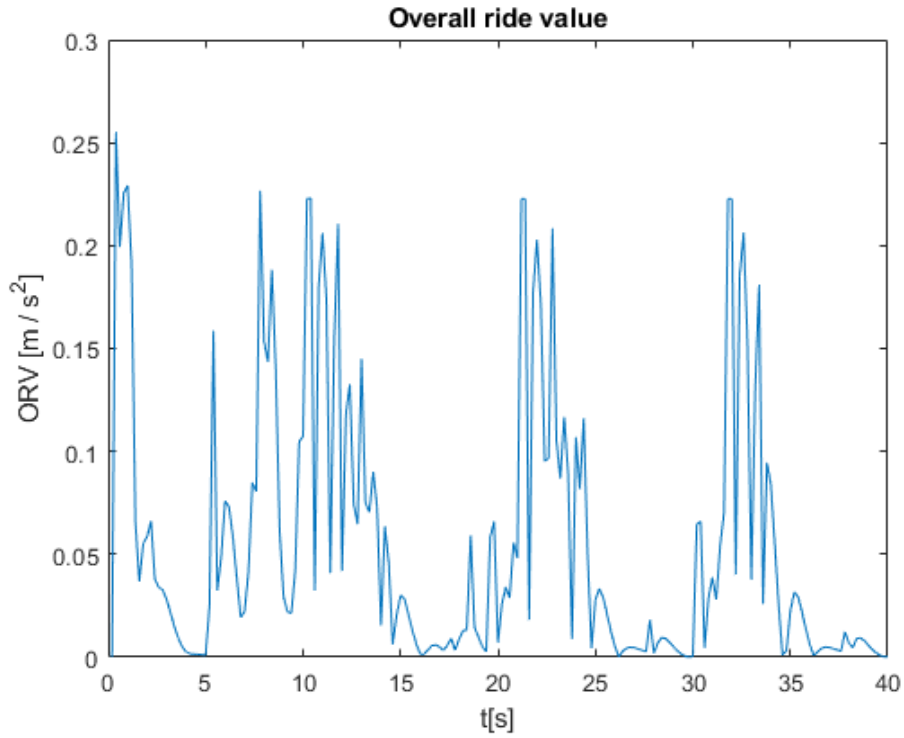


Figure 8.7: Overall ride value

As can be observed in the previous Figures (8.5, 8.6 and 8.7), the vehicle is able to stay within the admissible region even when the reference is set outside, so the validity of the position constraints is verified. Notice also how at the vertices of the octagon, the trajectory is not perfectly adjusted to the admissible region. This is due to ride comfort constraints, which restrict sharp changes on the acceleration.

8.4 Simple obstacle avoidance

The last requirement of this project is to guarantee obstacle avoidance while maintaining a comfortable ride from the point of the user. To test if this requirement is met, a first simulation with only one obstacle has been performed. The initial point and the reference are set so that the optimal trajectory would imply crossing through the obstacle, so that the avoidance of the object is forced.

As stated before, the angle of avoidance is chosen depending on orientation of the wheelchair. In the case of Figure 8.8, due to the previous trajectory, the following condition is satisfied:

$$\left\| \theta(k) - \left(\frac{h_{1st}}{h_{2st}} \right) \right\| \leq \frac{\pi}{2}$$

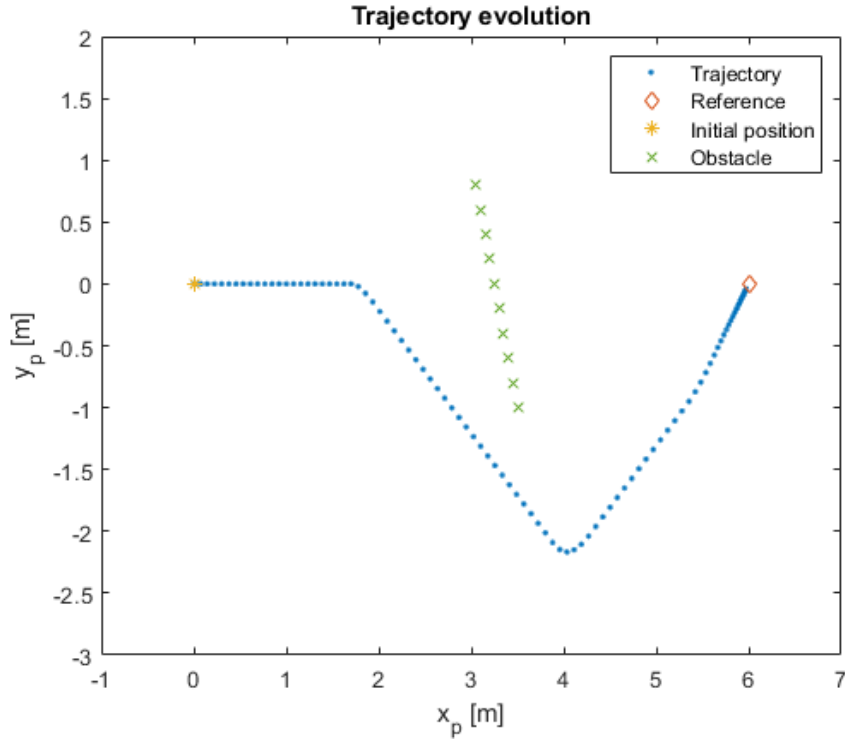


Figure 8.8: Trajectory with obstacle avoidance, same angle as the line

Therefore, the direction of the line defined by the obstacle is the direction utilized to perform the avoidance of the object.

In the case of Figure 8.9, again due to the trajectory followed before the activation of the vortex field, the following condition is satisfied:

$$\left\| \theta(k) - \left(\frac{h_{1st}}{h_{2st}} \right) \right\| \geq \frac{\pi}{2}$$

This results in the definition of the avoidance direction as the opposite direction of the line, which leads to an avoidance trajectory through the upper part of the obstacle.

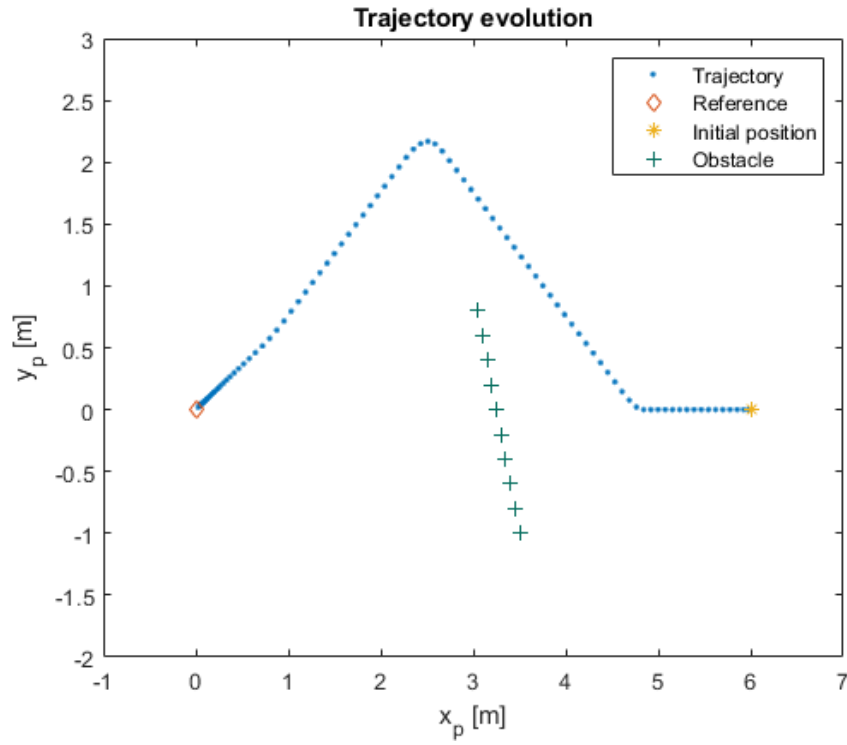


Figure 8.9: Trajectory with obstacle avoidance, opposite direction as the line

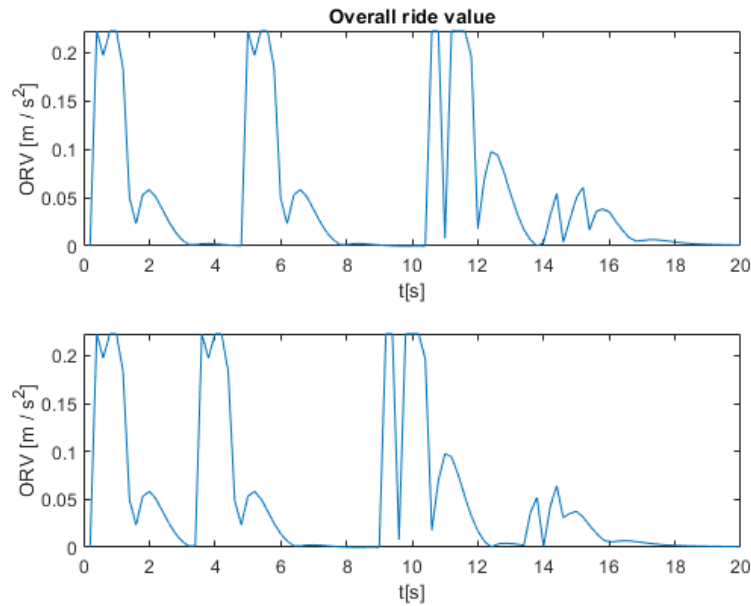


Figure 8.10: Overall ride value for the two cases. Upper data corresponds to the ORV of Figure 8.8, while the inferior plot corresponds to Figure 8.9

In view of the results showed, it is verified that the obstacle avoidance requirements with a comfortable ride are fulfilled.

8.5 Multiple object avoidance

An additional simulation has been carried through, in order to verify the ability of the vehicle to avoid multiple obstacles blocking the path to the reference. To do so, two obstacles have been generated, and the conditions of the simulation are:

$$\mathbf{P}_0 = (0, -0.5)$$

$$\mathbf{P}_{ref} = (9, -2)$$

As can be observed in Figures 8.11 to 8.13, the trajectory generated completely avoids the obstacles keeping a safe distance to them, and it is comfortable for the passenger as the values of ORV are always inferior to $ORV_{max} = 0.315m/s^2$

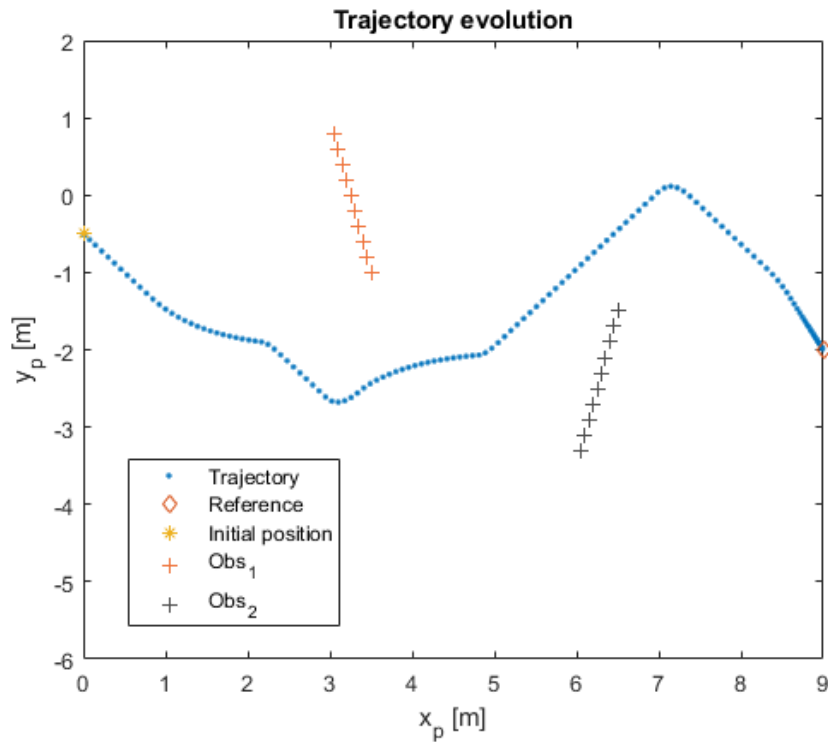
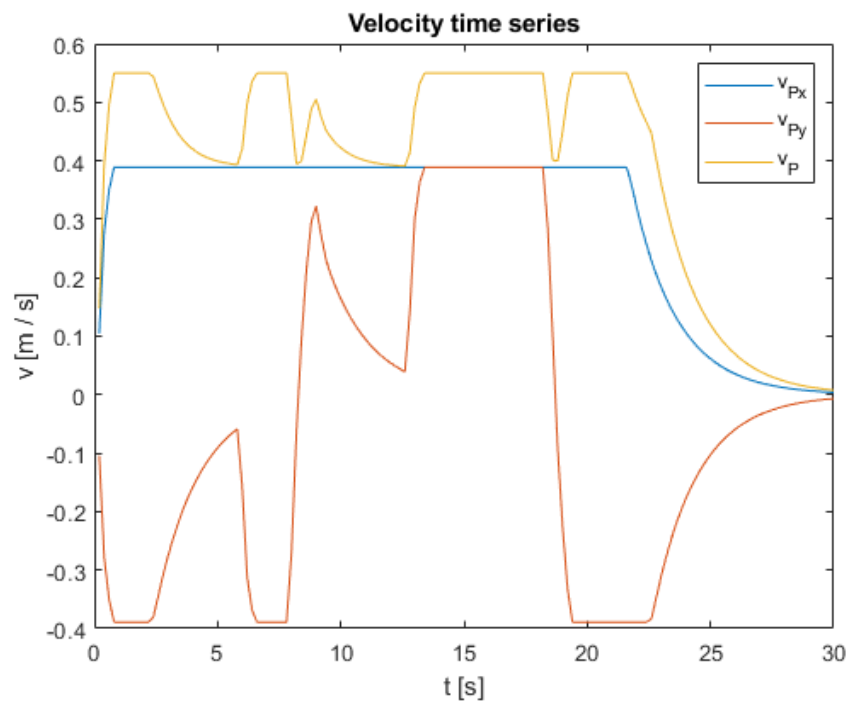
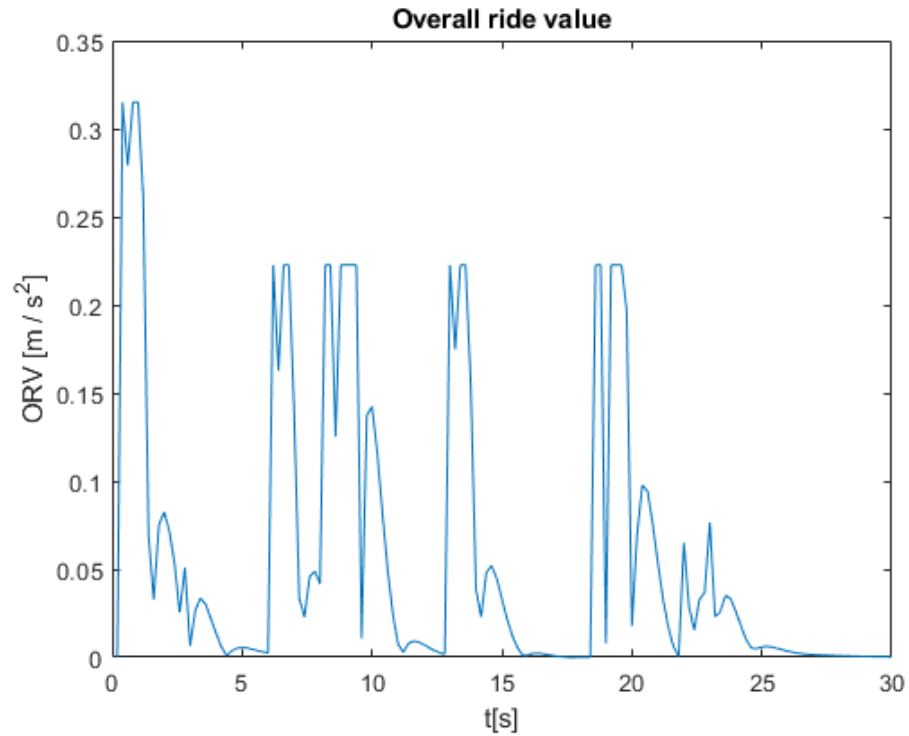


Figure 8.11: Obstacle avoidance with multiple obstacles

*Figure 8.12: Evolution of the velocity**Figure 8.13: Overall ride value*

Chapter 9

Conclusions and future developments

The main subject treated in this thesis has been the problem of the motion control of a wheelchair, which has been solved with the use of the Model Predictive Control technique. Parting from the work done in [1], a new control algorithm has been developed for assuring the comfort of the passenger.

As the main source of discomfort is related with vibrations, the acceleration of the wheelchair is weighted in the frequency domain to give more importance to the undesired vibrations related to certain frequencies, therefore being able to limit them. Additionally, obstacle avoidance requirements are also satisfied, guaranteeing a safe and comfortable experience for the passenger utilizing the autonomous wheelchair.

To prove the success of this approach, a large amount of tests have been performed in the form of simulations. The results obtained are satisfactory and demonstrate that the approach is viable. The next step would be to implement the resulting algorithm into the physical system, and test the performance on a real-world application.

Bibliography

- [1] Eugenio Ceravolo and Mauro Gabellone. “Controllo del moto di una sedia a rotelle autonoma mediante tecniche di controllo predittivo”. MA thesis. Politecnico di Milano, 2016.
- [2] M.J. Griffin. *Handbook of Human Vibration*. 988 p. Elsevier Science, 1996. ISBN: 9780123030412.
- [3] Yoichi Morales, Takahiro Miyashita, and Norihiro Hagita. “Social robotic wheelchair centered on passenger and pedestrian comfort”. In: *Robotics and Autonomous Systems* 87 (2017), pp. 355–362. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2016.09.010>. URL: <http://www.sciencedirect.com/science/article/pii/S092188901630570X>.
- [4] J. Urbano et al. “Velocity control of an omni-directional wheelchair considering user’s comfort by suppressing vibration”. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Aug. 2005, pp. 3169–3174. DOI: 10.1109/IR0S.2005.1545334.
- [5] R. W. Brockett. “Asymptotic stability and feedback stabilization”. In: *Differential Geometric Control Theory*. Birkhauser, 1983, pp. 181–191.
- [6] Simone Misiano. “Studio di Tecniche Distribuite di Controllo Predittivo Stocastico per l’Inseguimento di Segnali di Riferimento e Applicazione al Coordinamento di Robot Mobili”. MA thesis. Politecnico di Milano, 2014.
- [7] ISO 2631-1:1997 (E). *Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration*. Standard. Geneva, CH: International Organization for Standardization, 1997.
- [8] BS 6841:1987. *Guide to measurement and evaluation of human exposure to whole-body mechanical vibration and repeated shock*. Standard. London: British Standards, 1987.
- [9] M. S. Kim, K. W. Kim, and W. S. Yoo. “Method to objectively evaluate subjective ratings of ride comfort”. In: *International Journal of Automotive Technology* 12.6 (Dec. 2011), pp. 831–837. ISSN: 1976-3832. DOI: 10.1007/s12239-011-0095-8. URL: <https://doi.org/10.1007/s12239-011-0095-8>.